Luís Filipe Longo

Desenvolvimento de um aplicativo de análise de estruturas reticuladas planas em plataforma Android

Florianópolis

2015

Luís Filipe Longo

Desenvolvimento de um aplicativo de análise de estruturas reticuladas planas em plataforma Android

Trabalho de conclusão de curso submetido ao Departamento de Engenharia Civil da Universidade Federal de Santa Catarina para a obtenção do título de Engenheiro Civil

Universidade Federal de Santa Catarina – UFSC

Centro Tecnológico

Curso de Engenharia Civil

Orientador: Rafael Holdorf Lopez

Florianópolis

2015

Luís Filipe Longo
Desenvolvimento de um aplicativo de análise de estruturas reticuladas planas em plataforma Android/ Luís Filipe Longo. – Florianópolis, 2015-
256 p. : il. (algumas color.) ; 30 cm.

Orientador: Rafael Holdorf Lopez

Trabalho de conclusão de curso – Universidade Federal de Santa Catarina – UFSC
Centro Tecnológico
Curso de Engenharia Civil, 2015.

1. Análise estrutural. 2. Android. 2. Método dos deslocamentos. I. Rafael Holdorf Lopez. II. Universidade Federal de Santa Catarina. III. Graduação em Engenharia Civil. IV. Desenvolvimento de um aplicativo de análise de estruturas reticuladas em plataforma Android.

Luís Filipe Longo

Desenvolvimento de um aplicativo de análise de estruturas reticuladas planas em plataforma Android

Trabalho de conclusão de curso submetido ao Departamento de Engenharia Civil da Universidade Federal de Santa Catarina para a obtenção do título de Engenheiro Civil

Trabalho aprovado. Florianópolis, 24 de novembro de 2015:

Rafael Holdorf Lopez
Orientador

Prof. Ângela do Valle
Universidade Federal de Santa Catarina

André Ginklings da Cruz
Universidade Federal de Santa Catarina

Florianópolis
2015

Dedico este trabalho aos que passaram, aos que ficaram e, sobretudo, àqueles que estiveram comigo desde o início.

Agradecimentos

Agradeço em primeiro lugar a minha mãe, Clarice, mulher de resistente fibra cujo incentivo foi um dos grandes atores do meu desenvolvimento. Às poucas palavras que me restrinjo nunca serão o suficiente para expressar toda a minha gratidão pelas noites de vigília e pelos tão desmedidos cuidados.

Ao meu irmão Cadu, cuja diferença de idade nunca impediu que fôssemos grandes amigos – a decidir se por excesso de maturidade de sua parte ou por escassez da minha.

A toda minha família, que nunca falhou em me prover as melhores histórias e os melhores natais.

Aos meus amigos, aos quais peço que escusem a falta de nomes, afinal não poderia os citar. A todos vocês que me acolheram como em família, sem nunca esperar de mim mais que a amizade. Às noites de estudo que compartilhamos e aos cabelos que ficaram pelo caminho. Às festas memoráveis e às discussões impregnadas da melhor filosofia de bar que esta faculdade já conheceu. A todos vocês que, de um modo ou de outro, fizeram a minha passagem pela universidade mais inesquecível.

Por fim, ao meu orientador Prof. Dr. Rafael Holdorf Lopez, pela constante disponibilidade em dirimir as minhas dúvidas e me auxiliar com a parte teórica deste trabalho.

Do not go gentle into that good night,
Old age should burn and rave at close of day;
Rage, rage against the dying of the light.
(Dylan Thomas)

Resumo

A convergência para um mundo mais portátil é fato consumado no âmbito da engenharia. Isto se estende também para a esfera acadêmica, fazendo com que mais e mais estudantes recorram a dispositivos portáteis. O uso de tablets e smartphones, e a sua progressiva melhora, cria, então, uma demanda de programas que sejam compatíveis com essas plataformas. Nesse sentido, o presente trabalho tem o objetivo de prover uma alternativa de programa de análise estrutural para estruturas reticuladas planas, desenvolvendo um aplicativo para plataforma Android. É importante ressaltar, consequentemente, que este software não visa o uso profissional, salvo consultas rápidas e simplificadas, mas sim o uso em sala de aula. Levando em conta esta preferência, foram desenvolvidas ferramentas para facilitar este tipo de utilização, como a apresentação de elementos de cálculo – tal qual as matrizes de rigidez dos elementos. Também foi dada prioridade à facilidade de inserção, contando o programa com uma interface gráfica que mostrasse, concomitantemente, a inserção de novos elementos na estrutura, bem como a atribuição de carregamentos aos mesmos.

Palavras-chave: Análise Estrutural. Android. Método dos Deslocamentos.

Abstract

The convergence to a more portable world is an accepted fact amidst engineers. This extends as well to the academic sphere, resulting in an evergrowing number of students who resort to portable devices. The usage of tablets and smartphones, as well as their continuous improvement, creates, thus, a demand for programs that are compatible with these platforms. In this sense, the following work aims to provide an alternate option to the analysis of planed frame structures, developing an application for Android. It is important to highlight that, consequently, this software's goal is not professional use, unless for quick simplified verifications, but academic use. Considering this choice, tools to ease this kind of utilization were developed, like the display of calculation elements – such as element stiffness matrixes. The user-interface was also prioritized, resulting in a graphic entry system that shows, simultaneously, the insertion of new elements in the structures, as well as the designation of their loads.

Keywords: Structural Analysis. Android. Displacement Method.

Lista de ilustrações

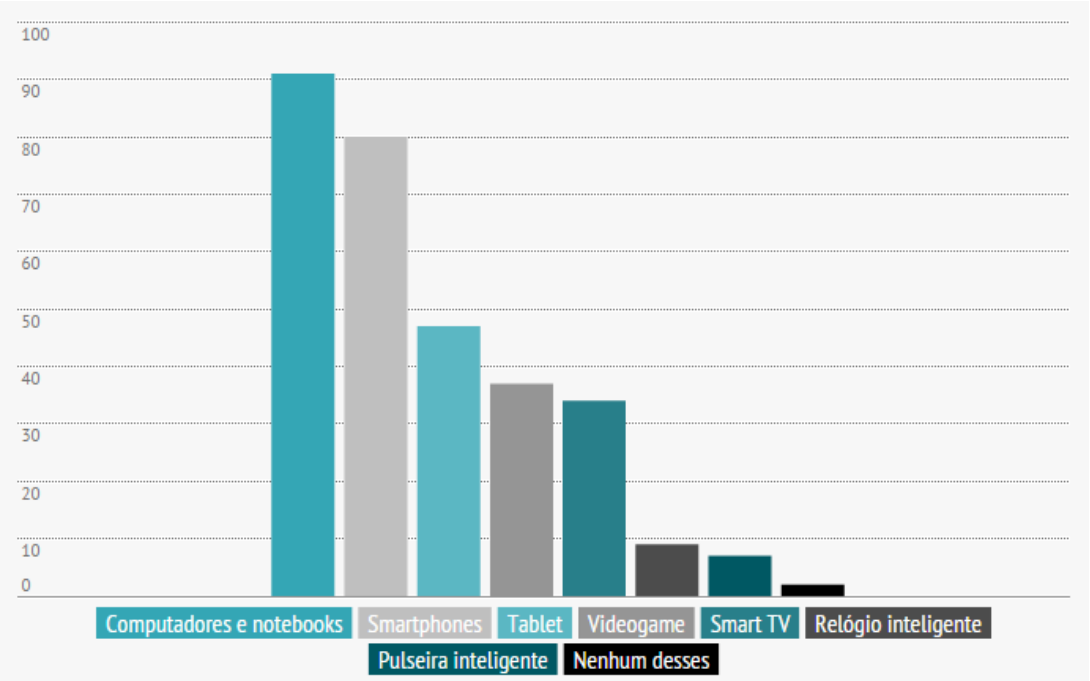
Lista de quadros

Sumário

1 Introdução

É inegável a participação de dispositivos portáteis no cotidiano da sociedade. De fato, segundo a GLOBAL WEB INDEX (c2005), cerca de 80% dos internautas mundiais utilizam *smartphones*. Este número, aliado aos 47% que utilizam *tablets*, acaba por abranger a quase totalidade dos usuários de internet (Figura 1).

Figura 1 – Uso de dispositivos móveis no mundo



Fonte: GLOBAL WEB INDEX (c2005)

Este fato é ainda mais exacerbado se considerarmos que *smartphones* e *tablets* são frutos de um passado recente, estabelecendo sua hegemonia em um período de tempo notavelmente curto. Tendo isto em mente, não é absurdo afirmar que tendência análoga se desenrole no âmbito educacional. De fato, ferramentas como esta facilitam em muito o processo de aprendizado, visto que adicionam uma nova dimensão a ele: o ensino deixa de ser estático para se tornar dinâmico e interativo. Ao encontro deste raciocínio, em pesquisa realizada em 2010 pelo instituto *Project Tomorrow*, demonstrou-se que existe um grande interesse da parte dos estudantes por novas tecnologias em sala de aula: "Estudantes querem poder interagir e aprender com a sua rede pessoal de especialistas usando ferramentas colaborativas e de comunicação de ponta" (Project Tomorrow (2009), tradução nossa).

Ademais, ruma-se para um novo tipo de ensino. Um ensino que não mais eleva o professor a uma posição de detentor de conhecimento, mas sim de tutor. O aprendizado, por sua vez, torna-se mais colaborativo – "[...] os estudantes começam a construir uma

rede personalizada de especialistas e, na verdade, também compartilham sua própria experiência do mesmo modo com outros" (Project Tomorrow (2009), tradução nossa).

Pode-se, portanto, traçar um perfil análogo no aprendizado da Engenharia Civil. Temos um processo de virtualização das tarefas já consolidado no mercado de trabalho, mas que não parece se estabelecer no âmbito do estudo - que, ainda segundo Project Tomorrow (2009), apresenta métodos mais conservadores de transmissão do conhecimento. Isto, unido ao fato de que os *softwares* mais populares voltados à engenharia são razoavelmente robustos, dificultando grandemente seu uso em sala de aula, faz com que o ensino perca uma grande oportunidade de se renovar, de integrar uma nova faceta, uma nova dimensão.

Neste sentido, este trabalho busca desenvolver, dentro dos limites estabelecidos em seu escopo, a portabilidade na área de análise estrutural, em engenharia civil. Para isto, escolheu-se realizar o desenvolvimento de um aplicativo em plataforma *Android*, cuja escolha deveu-se sobretudo à imensa popularidade do sistema e também a sua presença frequente em toda uma gama de aparelhos eletrônicos. Alternativamente, este trabalho também poderia ser realizado visando as demais plataformas voltadas a dispositivos portáteis, como *Windows Phone* ou *OSx*.Contudo, considerando o seu número restrito de usuários e alto custo de desenvolvimento, respectivamente, estas opções foram descartadas logo no estudo prévio.

Outro ponto que sobre o qual nos devemos ater neste primeiro momento são as limitações do aplicativo. Não configurando necessariamente um aspecto negativo, impor limitações na utilização de um produto desta sorte pode ajudar a definir melhor o seu escopo, além de assegurar que ele seja mais intuitivo para o usuário. Por conta disto, algumas linhas deverão ser traçadas para evitar a implementação de funções que terão pouca utilização no âmbito proposto pelo projeto. Como exemplo podemos citar que as barras deverão ter perfis constantes e material homogêneo elástico linear, ou então que as cargas distribuídas serão de distribuição linearmente variável, podendo apresentar valores iguais nas extremidades de uma barra.

Assim, tendo em vista o que até o momento foi exposto, visa-se que ao final deste trabalho, tenha-se um aplicativo para plataformas portáteis que disponibilize as ferramentas básicas de análise estrutural, tais como: inserção de elementos, cadastro de perfis, visualização de diagramas de esforços solicitantes, etc.

1.1 OBJETIVOS

Com o intuito de definir o escopo deste trabalho, deverão ser definidos também os objetivos deste. Se distinguirão entre um objetivo geral, englobando os aspectos holísticos do trabalho, e diversos objetivos específicos, que abarcarão aspectos mais pontuais do processo de desenvolvimento do aplicativo.

1.1.1 Objetivo Geral

Confeccionar um *software* portátil para plataforma *Android* de utilização intuitiva e facilitada e que possua todas as ferramentas necessárias para o estudo simplificado de uma estrutura plana porticada.

1.1.2 Objetivos Específicos

- Elaborar uma interface que permita a construção, edição e análise de resultados eficiente das três famílias de estruturas planas (barras, treliças e pórticos);
- Elaborar uma interface que permita uma navegação adaptada às necessidades do programa e também às possibilidades que o *touchscreen* oferece;
- Elaborar e implantar uma rotina baseada no método dos deslocamentos para cálculo matricial destas estruturas;
- Elaborar e implantar uma rotina para cálculo e desenho dos diagramas de esforços internos;
- Elaborar e implantar uma rotina para cálculo e desenho das deformações;
- Introduzir preferências pré-programadas, como perfis de uso trivial;
- Elaborar interfaces que permitam ao usuário o cadastro de seus próprios materiais e perfis;
- Implementar o cadastro de cargas linearmente distribuídas e pontuais.

1.1.3 Estrutura do Trabalho

Para um melhor entendimento de como este trabalho foi estruturado e também das sequentes etapas do processo de desenvolvimento do aplicativo, é interessante que se discorra brevemente sobre cada um dos capítulos que o compõe. Sendo assim, podemos listar:

- INTRODUÇÃO: Seção presente do trabalho, buscar dar ao leitor um panorama do trabalho a seguir.
- REVISÃO BIBLIOGRÁFICA: Nesta etapa serão estabelecidas todas as ferramentas necessárias para o desenvolvimento do aplicativo. Dentre elas, podemos destacar rotinas de implantação do método dos deslocamentos, equações para definir os diagramas de esforços internos, definição das reações de engastamento perfeito, etc.

- MÉTODOS: Esta seção descreverá sucintamente quais foram os recursos informáticos, vide programas e linguagens de programação, que foram utilizados durante o processo de desenvolvimento
- ESTRUTURAÇÃO DO APLICATIVO: Com todas as ferramentas, tanto matemáticas quanto informáticas, definidas, será possível dar início a esta seção, que desmembrará a estrutura do aplicativo. Aqui serão estabelecidas todas as classes e métodos programados, bem como as rotinas que foram utilizadas
- APRESENTAÇÃO DO APLICATIVO: Tendo em vista o conhecimento geral dos métodos expostos neste trabalho, optou-se por não resumí-lo a uma análise dos mesmos. Assim, é cabível que se estabeleça um manual de utilização do aplicativo, para o qual será utilizado um exemplo de pórtico.
- CONCLUSÕES: Por fim, serão apresentados algumas considerações sobre o trabalho, sobretudo no que tange as características finais do produto. Além disto, também serão sugeridos alguns temas para trabalhos vindouros.

2 REVISÃO BIBLIOGRÁFICA

Sussekind (1981) faz, no seu livro, uma breve definição de o que é uma estrutura no âmbito da engenharia, enumerando os elementos que a compõem e também explicando sinteticamente as restrições físicas que devem apresentar para que sejam consideradas como tais.

> "As estruturas se compõem de uma ou mais peças, ligadas entre si e ao meio exterior de modo a formar um conjunto estável, isto é, um conjunto capaz de receber solicitações externas, absorvê-las internamente e transmiti-las até seus apoios, onde estas solicitações externas encontrarão seu sistema estático equilibrante." (SUSSEKIND, 1981, pg. 1)

Desta definição, pode-se extrair algumas propriedades que caracterizarão o tipo de estrutura com que se está trabalhando e, consequentemente, qual o método mais adequado para a definição de seus esforços internos e externos. Seguindo a apresentação no parágrafo, temos que o primeiro parâmetro que deve ser definido são as propriedades geométricas dos elementos que constituem a estrutura. Esta resposta pode ser encontrada no próprio Sussekind (1981), que lista os elementos estruturais como:

a) possuindo uma dimensão preponderante às outras duas;

b) possuindo duas dimensões preponderantes à terceira;

c) não possuindo nenhuma preponderância de dimensões;

Este tipo de classificação pode dividir, segundo o autor, as estruturas entre reticuladas, de placas e de blocos, caracterizando as primeiras a grande maioria das estruturas triviais.

Seguindo, vemos que outra característica levantada pelo autor é o modo com que as cargas são aplicadas na estrutura. Esta informação é essencial, visto que definirá o tipo de esforços internos presentes nos elementos e, por consequência, quais esforços poderão ser negligenciados e quais deverão ser levados em consideração. O modo com que a estrutura é carregada, aliado com o modo com que os seus elementos estão solidarizados entre si, irá ditar qual o tipo de modelo utilizado, podendo levar a um modelo de viga, pórtico, treliça, placa, etc.

Por fim, percebe-se que o autor também se preocupa em definir um sistema estático equilibrante, que se refere às condições de apoio físico da estrutura. Verificar como a

estrutura está restringida também torna-se importante, pois tem impacto direto no método usado para a definição de seus esforços internos e externos. Corrobora com isto Martha (2010), quando diz que "Em geral, as equações de equilíbrio fornecem condições necessárias, mas não suficientes para a determinação dos esforços no modelo estrutural" (MARTHA, 2010, pg. 20). Aí definem-se dois tipos de estruturas: as isostáticas, cuja resolução pode ser obtida unicamente a partir das condições equilíbrio, e as hiperestáticas, cuja solução demanda condições suplementares de compatibilização de deformações e deslocamentos. É interessante notar também que o autor evidencia a natureza mais trivial das estruturas hiperestáticas, classificando as condições de equilíbrio como essenciais, mas não suficientes.

Como terceiro tipo de estrutura, temos ainda as hipostáticas, para as quais o grau de restrição não é suficiente para absorver o carregamento atuante. Estas estruturas não poderão ser analisadas por este método, visto que não satisfazem as equações essenciais da estática, apresentando comportamento dinâmico (SUSSEKIND, 1981).

2.1 MODELOS DE CÁLCULO

"Dois métodos diferentes podem ser utilizados para analisar estruturas matricialmente: o método das forças e o método dos deslocamentos" (KASSIMALI, 2012, pg. 4, tradução nossa). Martha (2010) pontua que o primeiro busca determinar um conjunto solução de forças que respeite as condições de equilíbrio e atenda também às condições de compatibilidade. Já o segundo, como explica o autor, busca um conjunto solução de deslocamentos que respeite as condições de compatibilidade e atenda também às condições de equilíbrio.

Kassimali (2012) ainda adiciona, sobre o uso destes métodos: "O método dos deslocamentos é mais sistemático e pode ser implementado mais facilmente em computadores, sendo preferido para a análise de estruturas maiores e mais hiperestáticas" (KASSIMALI, 2012, pg. 5, tradução nossa). Sobre isto, Martha (2010) contribui, explicitando que o método das forças dispõe de uma metodologia que dificulta a sua implementação, o que o torna uma escolha pouco trivial no desenvolvimento de ferramentas computacionais.

Levando em consideração o posicionamento dos dois autores, fica claro que o método mais indicado a ser implementado é o método dos deslocamentos. Mais especificamente, buscaremos definir um algoritmo simples de cálculo baseado no método da rigidez direta, derivado deste anterior, que apresenta uma estrutura propícia para este tipo de aplicação (MARTHA, 2010).

2.2 O MÉTODO DOS DESLOCAMENTOS

2.2.1 Filosofia do método

Como dito anteriormente, o método dos deslocamentos se diferencia do método das forças por possuir uma formulação muito mais sistemática, sendo que "[...] inicia a resolução da estrutura pela determinação dos seus esforços para, a partir deles, obter deformações." (SUSSEKIND, 1987, pg. 1). "As incógnitas deste método serão, então, os ângulos de rotação e os deslocamentos lineares sofridos pelos nós das diversas barras." (SUSSEKIND, 1987, pg. 1). Assim, para que possamos definir os esforços internos e externos de uma estrutura, será necessário que definamos previamente quais são os deslocamentos causados pelos carregamentos solicitantes. Para fins de definição, representaremos cada um destes deslocamentos segundo a notação 2.1, com i representando seus respectivos índices:

$$D_i \tag{2.1}$$

Para determinar estes valores, poderemos nos valer do princípio da superposição:

> "O princípio da superposição afirma que a tensão ou o deslocamento resultante no ponto pode ser determinado se antes se determinar a tensão ou o deslocamento causado por cada componente da carga agindo separadamente sobre o elemento." (BEER et al., 2010, tradução nossa)

Estendendo essa definição ao método, nos serve Martha (2010):

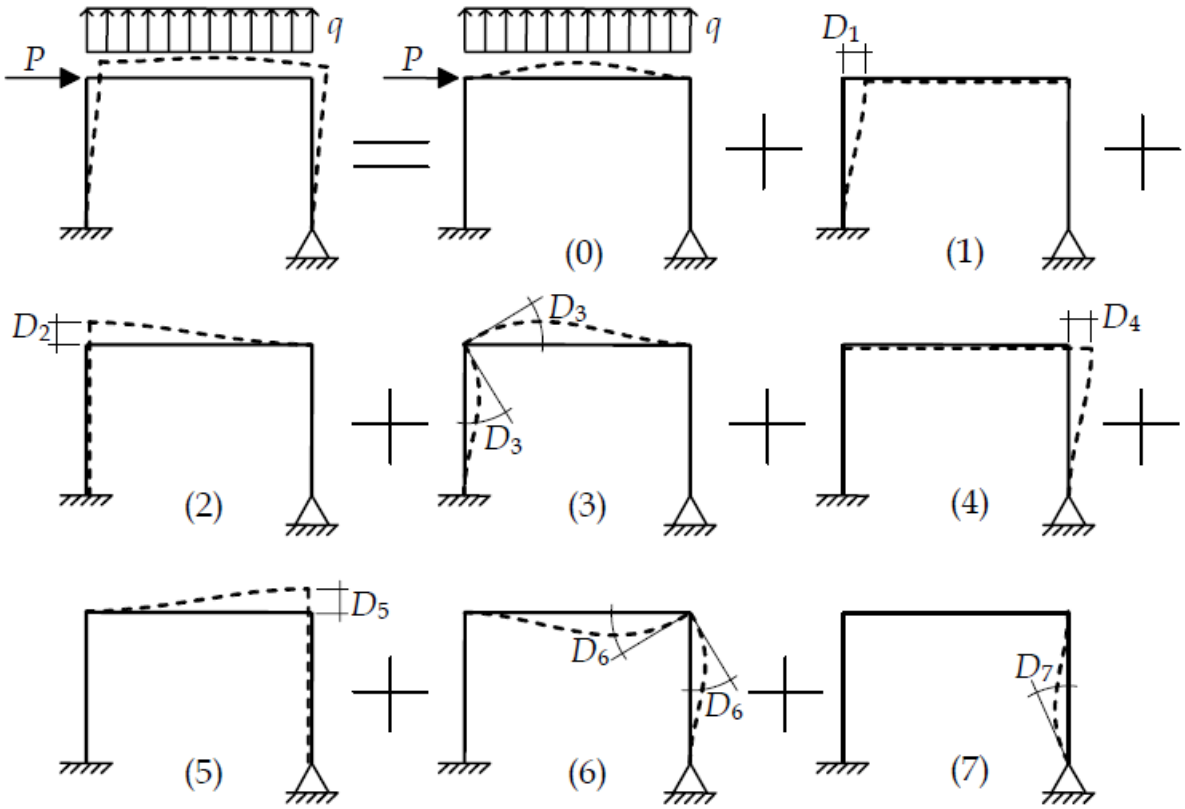
> "Somar uma série de soluções básicas (chamadas de casos básicos) que satisfazem as condições de compatibilidade, mas que não satisfazem as condições de equilíbrio da estrutura original, para na superposição restabelecer as condições de equilíbrio." (MARTHA, 2010, pg. 193)

Isto implica que podemos modelar uma determinada estrutura estaticamente indefinida como "[...] uma superposição de soluções cinematicamente definidas, isto é, de configurações de deformadas conhecidas" (MARTHA, 2010). Refraseando, é possível decompor o sistema estrutural principal, com seus carregamentos, em diversos outros casos, de modo que com a sua superposição seja possível definir a deformada deste sistema principal. Graficamente, podemos representar este princípio conforme a figura 2.

Para que isto seja possível, contudo, é importante que algumas restrições sejam respeitadas. Para definir estas hipóteses, podemos consultar Ghali et al. (2009), quando diz que "[...] quando as deformações na estrutura são proporcionais às forças aplicadas o princípio da superposição é válido" (GHALI et al., 2009, pg. 92, tradução nossa). Ainda

nas palavras do mesmo autor “Uma estrutura pode se comportar não linearmente caso as cargas aplicadas causem grandes modificações na geometria” (GHALI et al., 2009, pg. 93, tradução nossa).

Figura 2 – Configuração deformada de um pórtico plano formada pela superposição de configurações deformadas elementares.



Fonte: Martha (2010)

Analisando a figura 2, podemos perceber que de fato uma estrutura com seus devidos carregamentos pode ser desmembrada em diversos outros casos. Seguindo o esquema, vemos que o caso (0) se refere às ações externas atuando sobre as barras e nós deste pórtico. Nesta configuração, admite-se que os nós não apresentam deslocamentos lineares ou rotações, sendo que as deformações se limitam aos elementos de barra. Estes deslocamentos, desprezados no caso (0), serão levados em consideração nos casos de (1) a (7), cada um deles sendo referente a um deslocamento distinto.

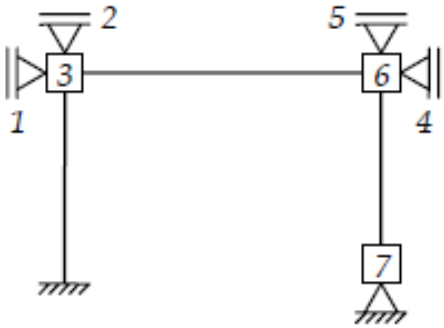
2.2.2 Sistema Hipergeométrico

Para que estes casos possam ser representados, podemos nos valer de uma estrutura fictícia, denominada sistema hipergeométrico, ou SH (Figura 3). Este mantém as características da estrutura inicial, mas tem todos os seus nós completamente engastados, para isto estabelecendo *apoios fictícios* (MARTHA, 2010). Estes engastes permitem que cada uma das deslocabilidades D_i da estrutura seja isolada, sem que uma tenha influência

sobre as demais. Todavia, por não possuir nós livres, suas deslocabilidades não poderão ser originadas da aplicação de uma carga, mas sim do recalque destes apoios fictícios.

Indicados na figura 3, estão representados os apoios fictícios que foram adicionados à estrutura para que se adequasse ao modelo hipergeométrico. A cada um deles foi atribuído um índice, que corresponde também ao índice da deslocabilidade gerada com o seu recalque. Sendo assim, se recalcarmos o apoio 1, obteremos o caso (1) representado na figura 3 e consequentemente a deslocabilidade D_1. O mesmo vale para os demais apoios, totalizando sete deslocabilidades, número que está de acordo com o inicialmente definido.

Figura 3 – Sistema hipergeométrico do pórtico da figura 2



Fonte: Martha (2010)

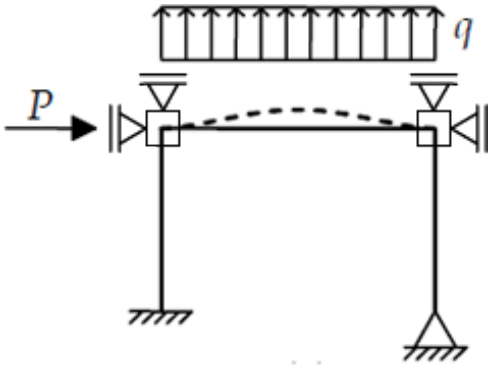
Para dar seguimento à resolução da estrutura, é necessário que se defina como serão calculados o casos nos quais a decompusemos. Ainda levando em consideração o pórtico apresentado, dividiremos este cálculo em duas etapas: em um primeiro momento, serão calculados os esforços causados pelo carregamento na estrutura, isto é, o caso (0). Feito isto, será apresentado o modelo de cálculo para os demais casos, sendo que o caso (3) será tomado como base, a mero título de exemplo.

2.2.2.1 Solução para o caso (0)

Quando visualizamos as cargas aplicadas sobre o SH (Figura 4), logo percebemos que podemos decompô-lo no cálculo de várias barras. Isto ocorre pois, como todos os nós se encontram engastados, não há transferência de esforços de uma barra para outra. Sendo assim, para determinar as reações desta estrutura, basta definir as reações em cada uma das barras que a compõem.

A estes esforços, damos o nome de forças de engastamento perfeito, denotados também por F_{i0}. Esta notação segue a mesma estrutura utilizada para as deslocabilidades, o que implica que o valor de F_{i0} se refere à reação no apoio fictício associado com D_i quando aplicados os carregamentos inerentes ao caso 0. A determinação dos valores destes esforços poderá ser realizada através de outros métodos de análise de estruturas hiperestáticas, sendo conveniente o uso do método da linha elástica, visto que se trata de uma barra.

Figura 4 – Caso 0



Adaptado de: Martha (2010)

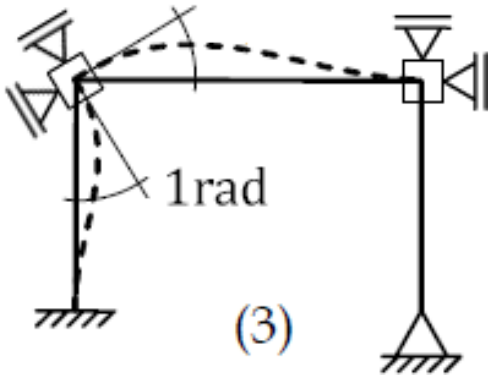
Todavia, este processo será tratado mais a frente, para que já haja certo conhecimento do método e se possa fazer mais algumas ressalvas ao cálculo.

2.2.2.2 Solução para os demais casos

Contrariamente ao caso (0), nos demais casos da metodologia, não são cargas que causarão as reações nos apoios fictícios, mas sim o recalque de algum destes apoios. Sendo assim, o cálculo das reações deverá partir da avaliação dos efeitos destas deformações no SH. Admitiremos, para cada um dos casos, um deslocamento unitário nos apoios fictícios, de modo que os efeitos causados por este recalque sejam multiplicados pelo valor de D_i, que será definido posteriormente no método (MARTHA, 2010).

Neste intuito, podemos observar a figura 5: com o recalque do apoio fictício 3, surgirão esforços nos demais apoios para equilibrá-lo, de modo que as condições de deformada impostas pelo SH sejam respeitadas (SUSSEKIND, 1987). Cabe ainda ressaltar que estas condições caracterizam-se em deslocamentos nulos nos demais apoios, visto que apenas o apoio 3 sofreu recalque.

Figura 5 – Caso 3



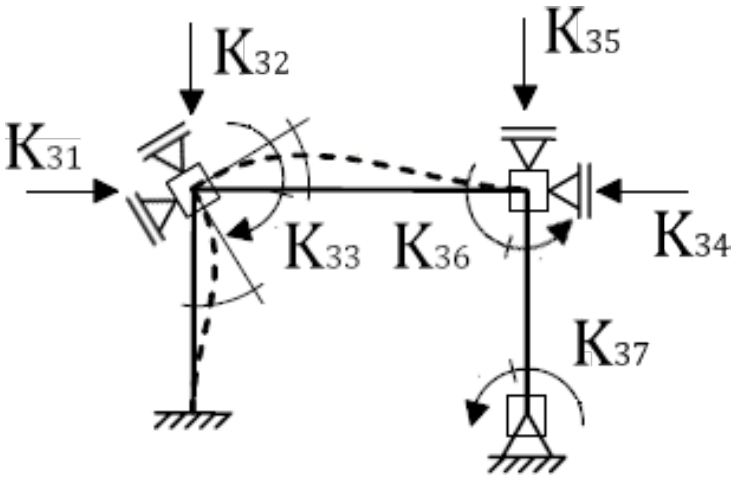
Adaptado de: Martha (2010)

Os efeitos que equilibram a estrutura deformada são conhecidos como coeficientes de rigidez global da estrutura, visto que indicam a capacidade que ela tem de se manter

na mesma posição perante os carregamentos impostos. Assim, quanto maior este valor, menor serão os deslocamentos identificados em cada um dos pontos. Estes coeficientes serão denotados por K_{ij}, onde i se refere ao apoio fictício que foi deslocado, neste caso o 3, e j ao apoio fictício onde foram verificados os efeitos.

No caso particular deste pórtico, o índice j variará de 1 à 7, sendo que cada um dos coeficientes (K_{31}, K_{32}, ..., K_{37}) a ele inerentes estão representados na figura 6. Outro ponto que deve ser levantado é que a unidade de medida destes coeficientes corresponde à unidade da força, ou momento, dividida pela unidade do deslocamento (MARTHA, 2010). Assim, K_{31} será medido em $\frac{KN}{m}$, por exemplo, enquanto K_{33} será medido em $\frac{KNm}{rad}$.

Figura 6 – Coeficientes de rigidez global para o caso 3.



Adaptado de: Martha (2010)

Como podemos ver, esta análise se mostra simples para estruturas de composição descomplicada, mas quando atingem níveis maiores de complexidade, este tipo de abordagem pode se tornar ineficaz. Assim, como na seção anterior, a determinação dos valores destes coeficientes será postergada, de modo que seja possível fazer algumas considerações adicionais antes de determinar ditas equações.

2.2.3 Articulações Internas

Articulações internas são ligações entre elementos de uma determinada estrutura que permitem algum tipo de deslocamento entre eles, tendo, por característica, não contribuir com a rigidez associada à deslocabilidade sobre a qual agem. Como exemplos, podemos citar rótulas, nós semirrígidos e vínculos deslizantes, sendo as primeiras mais abordadas durante a graduação e as únicas que serão estudadas por este trabalho.

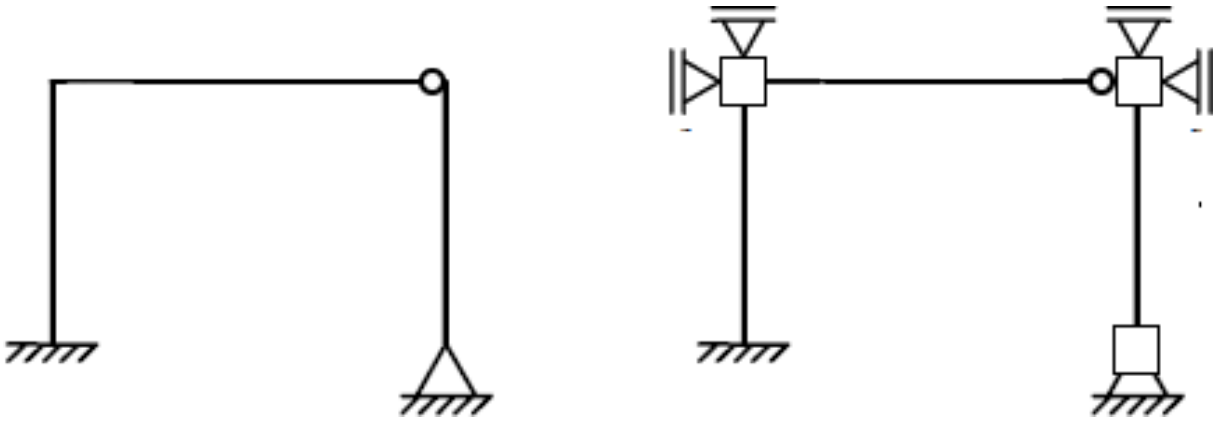
A aplicação de rótulas assegura a liberação da continuidade da rotação (MARTHA, 2010), isto é, permite que dois elementos adjacentes por ela conectados apresentem rotações divergentes. É importante notar que as rótulas também não transmitirão momentos entre estes elementos, assegurando apenas a transmissão de forças (SUSSEKIND, 1981). Por

consequência, dado que não haja nenhuma carga de momento aplicada no ponto rotulado, o diagrama de momentos fletores em dito ponto será sempre igual a zero.

Como é possível constatar, a aplicação de uma articulação diferenciada na estrutura modifica a distribuição de esforços na mesma, sendo necessário que seja levada em consideração no modelo do SH. Todavia, previamente a estabelecer um modelo que compreenda estes elementos, devemos analisar as duas maneiras sob as quais eles podem estar presentes na estrutura.

Em princípio, podemos considerá-las aplicadas sobre as extremidades dos elementos que compõem a estrutura. Nestes casos, o SH é de fácil dedução, sendo que para considerar os efeitos da articulação interna, basta aplicá-la no elemento em questão no SH (MARTHA, 2010), como mostra a figura 7.

Figura 7 – Pórtico com articulação interna na extremidade de um elemento



Adaptado de: Martha (2010)

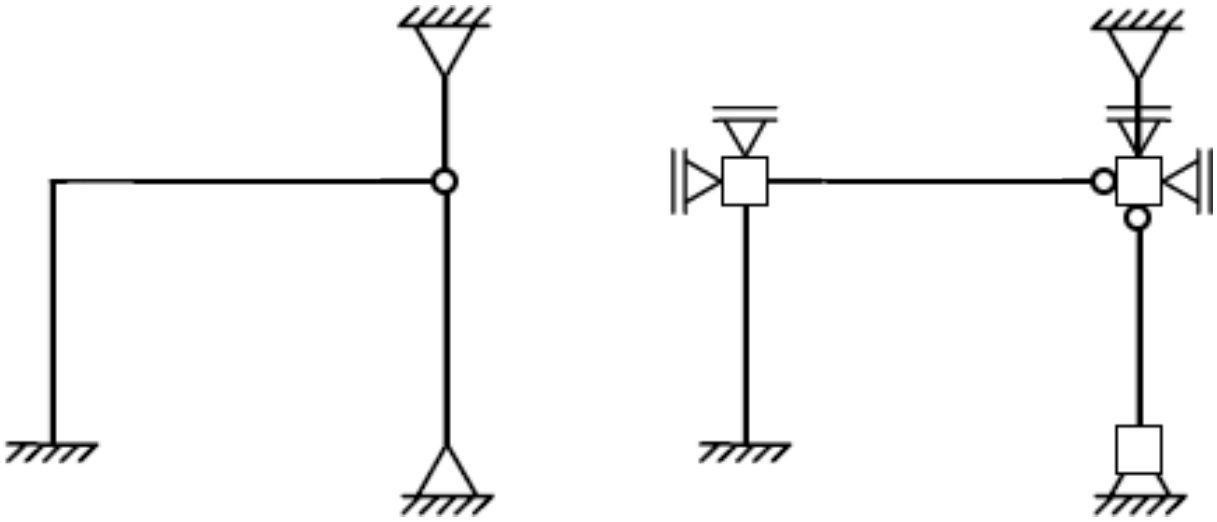
Adicionalmente, ainda podemos considerar a vinculação como aplicada no nó. Nestes casos, todos os elementos ligados a este nó estarão rotulados na extremidade em questão. Todavia, durante a análise, caso a rótula seja aplicada em cada um destes elementos, tem-se que a estrutura se tornará hipostática. (KASSIMALI, 2012). Neste sentido, o autor fornece uma possível solução:

> "Talvez o modo mais direto e eficiente de contornar esta dificuldade é eliminar os graus de liberdade rotacionais das articulações rotuladas da análise, do modo a modelar uma das ligações como restritas (ou fixadas) em relação à rotação." (KASSIMALI, 2012, pg. 352, tradução nossa)

Sendo assim, ao menos uma das barras que chegam ao nó deverá ser considerada como engastada na modelagem do SH (Figura 8). Esta ferramenta tem uma implicação em especial que deverá ser levada em conta na análise da estrutura: caso haja alguma carga pontual de momento aplicada neste ponto específico, a barra cuja articulação

foi considerada como engastada absorverá esta solicitação. Logo, temos que no pórtico representado pela figura 8, o tramo que absorverá uma possível carga de momento aplicada sobre o nó será o superior, que não apresenta a sua extremidade rotulada.

Figura 8 – Pórtico com articulação interna em um nó



Adaptado de: Martha (2010)

De todo modo, em ambos os casos, quatro situações de rotulação são possíveis para cada um dos elementos. Supondo uma barra, é possível que ela apresente ambas as articulações rígidas, uma das extremidades rotuladas (aí contabilizando duas possibilidades), ou então ambas as extremidades rotuladas. Para identificar mais facilmente a qual situação está submetido cada membro, é cabível estabelecer um índice para cada um destes (Quadro 1), assim como sugerido por Kassimali (2012).

Quadro 1 – Tipo do membro em relação a sua rotulação

Tipo do membro (MT)	Nó inicial (N_0)	Nó final (N_1)
0	Engastado	Engastado
1	Rotulado	Engastado
2	Engastado	Rotulado
3	Rotulado	Rotulado

Fonte: autor

2.2.4 Compatibilidade Estática

Uma vez determinados os efeitos causados pelas cargas em cada um dos apoios e também os coeficientes de rigidez global da estrutura, é necessário que se faça a compatibilização estática da estrutura. Para isto, podemos observar a estrutura original estudada. Por ela não possuir os apoios fictícios impostos no SH, vem que as deslocabilidades associadas a estes apoios deverão ser tais que não existam reações estáticas finais neles. Tendo esta condição sido alcançada, a superposição dos casos estabelecidos "[...] reproduzirá fielmente o comportamento elástico e estático da estrutura dada" (SUSSEKIND, 1987, pg. 24).

Tomando esta condição, podemos enunciar a seguinte relação (Equação 2.2) para uma deslocabilidade D_i qualquer:

$$F_i + \sum_{j=1}^{n} K_{ij} \cdot D_i = 0 \tag{2.2}$$

, onde n representa o número de deslocabilidades da estrutura em questão. No caso do pórtico exemplo, $n = 7$. A partir da equação 2.2, podemos obter um sistema de equações para o pórtico exemplo, como mostrado na equação 2.3:

$$\begin{cases} F_{10} + K_{11} \cdot D_1 + K_{12} \cdot D_2 + ... + K_{1n} \cdot D_n \\ F_{20} + K_{21} \cdot D_1 + K_{22} \cdot D_2 + ... + K_{2n} \cdot D_n \\ \vdots \\ F_{n0} + K_{n1} \cdot D_1 + K_{n2} \cdot D_2 + ... + K_{nn} \cdot D_n \end{cases} \tag{2.3}$$

A solução do sistema de equações 2.3 permite determinar os valores de D_i, o que permitirá posteriormente determinar também as reações da estrutura em seus apoios. Além disso, por se tratar de um sistema de equações, pode ser escrito em sua forma matricial, de modo que assume a forma representada na equação 2.4:

$$\begin{Bmatrix} F_{10} \\ F_{20} \\ \vdots \\ F_{n0} \end{Bmatrix} + \begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix} \cdot \begin{Bmatrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{Bmatrix} = 0 \tag{2.4}$$

, ou ainda, em formato reduzido (Equação 2.5):

$$\{F\} + [K] \cdot \{D\} = 0 \tag{2.5}$$

, onde denominamos:

$\{F\}$: vetor global das forças de engastamento perfeito;

[K] : matriz global de rigidez da estrutura;

$\{D\}$: vetor global de deslocamentos da estrutura.

Os métodos de solução deste sistema serão abordados mais à frente, visto que também é necessário definir uma rotina de solução passiva de implementação computacional.

A matriz de rigidez da estrutura possui algumas propriedades que são dignas de nota. Segundo o Teorema de Maxwell, "Em uma estrutura linear elástica, o deslocamento em uma coordenada i devido a uma força unitária em uma coordenada j é igual ao deslocamento em j devido a uma força unitária agindo em i." (GHALI et al., 2009, pg.262, tradução nossa). Salvo demonstração, o caminho inverso também se verifica, o que implica que uma força em i causada por um deslocamento unitário em j será igual a uma força em j causada por um deslocamento unitário imposto em i. Sendo assim, podemos dizer que:

$$K_{ij} = K_{ji} \tag{2.6}$$

Isto implica que a matriz de rigidez será, via de regra, uma matriz simétrica, podendo ser representada como segue:

$$K = \begin{bmatrix} K_{11} & K_{12} & K_{13} & \dots & K_{1n} \\ & K_{22} & K_{23} & \dots & K_{2n} \\ & & K_{33} & \dots & K_{3n} \\ & & & \ddots & \vdots \\ & & & & K_{nn} \end{bmatrix} \tag{2.7}$$

Outro ponto que merece destaque nas propriedades da matriz de rigidez é o fato de que, para estruturas estáveis, sejam elas iso ou hiperestáticas, ela será sempre inversível, portanto não singular. Esta propriedade pode ser deduzida de Ghali et al. (2009), quando diz que: "[...] o pórtico somente será estável se a sua matriz de rigidez for definida positiva e, logo, seu determinante maior que zero".

Aliada a esta propriedade, vem da álgebra linear que um sistema somente será compatível e determinado (ou seja, terá uma única solução) caso a sua matriz de coeficientes puder ser reduzida à matriz identidade, logo, terá de ser inversível (BOLDRINI et al., 1980). Disto sai que, caso o sistema de equações definido em 2.3 apresente várias soluções, a estrutura será caracterizada como hipostática.

2.2.5 Convenção de sinais

Por fim, se faz necessário que determinemos, para cálculos futuros, qual a convenção de sinais que será utilizada, tanto para forças quanto para deslocamentos. Sistematicamente,

a convenção demonstrada pelo quadro 2, elaborado com base no escrito por Ghali et al. (2009) será adotada.

Quadro 2 – Convenção de sinais para o método dos deslocamentos

	+	−
Deslocamentos		
Horizontais	→	←
Verticais	↑	↓
Rotações	↪	↩
Forças		
Horizontais	→	←
Verticais	↑	↓
Momentos	↪	↩
Esforços Internos		
Normais	←□→	→□←
Cortantes	↑□↓	↓□↑
Fletores	↩□↪	↪□↩

Fonte: autor.

2.3 O método dos deslocamentos sob uma perspectiva computacional

O pórtico utilizado como exemplo tem uma geometria simples, sendo que a determinação das variáveis envolvidas no métodos dos deslocamentos não é laboriosa. Contudo, quando a geometria do pórtico em questão se torna mais complexa, englobando elementos inclinados, ou de seções e materiais diferentes, precisar os coeficientes de rigidez e as forças de engastamento perfeito assume um caráter mais árduo, tornando sua implementação computacional mais sinuosa.

Destarte, convém segmentar a estrutura, dividindo seus elementos de modo que cada um seja analisado separadamente e que as suas contribuições para o vetor de forças de engastamento perfeito e a matriz de rigidez sejam posteriormente somados.

Nas palavras de Kassimali (2012), esta segmentação requer que se definam algumas peculiaridades:

> "Visto que é conveniente expressar as relações básicas de elasticidade dos membros em termos das forças e deslocamentos paralelos e perpendiculares a estes membros, é definido um sistema de coordenadas locais para cada membro da estrutura." (KASSIMALI, 2012, pg. 50, tradução nossa)

Logo, possuiremos dois sistemas de coordenadas: um posicionado em relação à estrutura como um todo, denotado sistema global, e outro que será posicionado em cada um dos elementos, cuja origem estará localizada no seu nó inicial N_0, denotado sistema local.

Neste último, poderemos analisar mais eficientemente as relações forças-deslocamento, sendo possível definir uma rotina de cálculo mais sistemática, visto que o elemento de estudo será sempre uma única barra. Uma vez que as variáveis inerentes às barras tenham sido determinadas, nos valeremos de relações de correspondência entre os sistemas global e local para anexar as contribuições de cada elemento na equação global da estrutura.

Para que possamos fazer isto, será necessário analisar a estrutura como não restringida, isto é, supõe-se, para a montagem da equação 2.5 que a estrutura não possui apoios. Isto faz com que seja necessário analisar os nós da estrutura para todas as suas possíveis deslocabilidades, não somente as carentes de restrição. Posteriormente, para a solução da equação 2.5, consideraremos a estrutura já restringida, de modo que poderemos eliminar as equações associadas aos deslocamentos restritos. Isto faz com que a análise seja mais direta, visto que enquanto as barras são estudadas em separado não haverá necessidade de se atentar às condições de apoio da estrutura.

Assim, cabe, em um primeiro momento, estudar o equacionamento destas barras, para que se possa, em seguida, deduzir as relações de correspondência entre os dois sistemas de coordenadas. Tendo adotado uma estrutura porticada, temos que cada barra possuirá seis deslocabilidades, referentes aos deslocamentos axiais, transversais e rotacionais para cada um de seus nós (Figura 9), o que implica em que para cada um dos membros da estrutura será necessário determinar seis forças de engastamento perfeito e uma matriz de rigidez de dimensões $[6 \times 6]$.

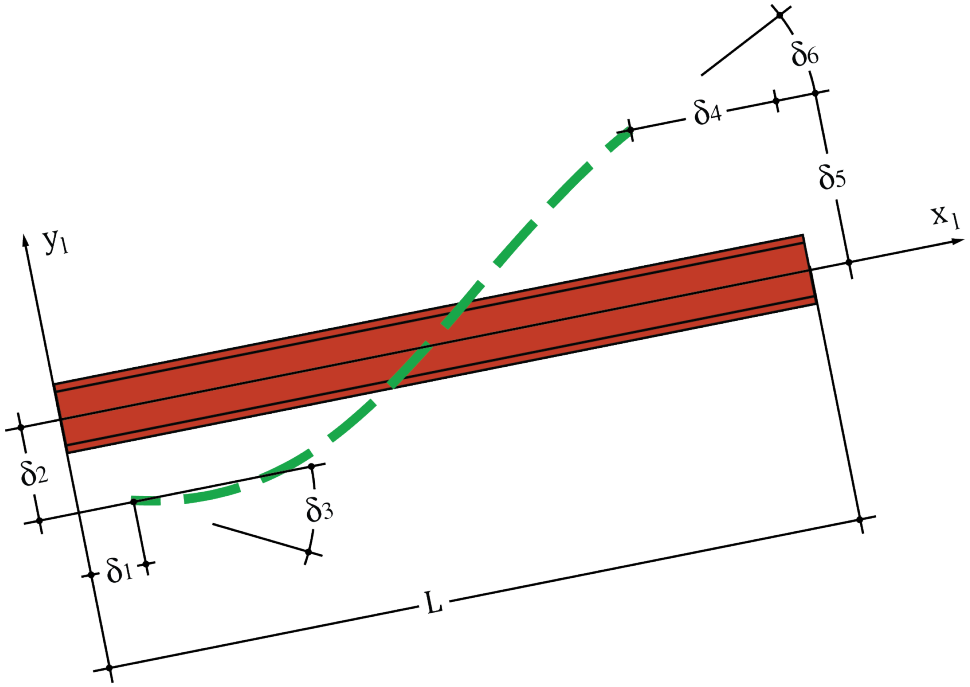
2.3.1 Forças de engastamento perfeito no sistema local

Para o cálculo das forças de engastamento perfeito, deveremos fazer algumas considerações. Deve ficar explícito que estas estão ligadas a características do programa a ser desenvolvido, não estando relacionadas a limitações do método de cálculo.

No caso das barras, serão consideradas como homogêneas e prismáticas. Isto significa que não haverá mudança de seção ou de materiais ao longo do comprimento das mesmas. Além disso, admite-se que estas poderão estar submetidas a quatro configurações de articulação, apresentando valores diferentes para as forças de engastamento perfeito em cada uma delas.

Ademais, deveremos também definir a que tipo de cargas distribuídas a estrutura estará submetida. Neste trabalho nos limitaremos a barras submetidas a cargas distribuídas linearmente variáveis, que poderão ser definidas pelo vetor de componentes x e y. A

Figura 9 – Barra com suas respectivas deslocabilidades no sistema local



Fonte: autor.

caracterização desta carga pode ser encontrada no item A.1 do respectivo apêndice.

Já a aplicação de eventuais cargas concentradas deverá ser feita nos nós da estrutura, sendo que definiremos um vetor exclusivo a elas nas próximas seções. Esta separação faz com que seja mais fácil tratar os casos de carregamento, não sendo necessário desenvolver uma rotina de cálculo dos esforços causados para cada uma das possíveis forças aplicadas nas barras.

Tendo feito estas ressalvas, podemos partir para a determinação das forças de engastamento perfeito. Por se tratarem de forças locais, elas serão representadas por f_i0. Como exposto acima, estes valores deverão ser determinados para quatro casos, adotando as configurações de articulação em cada uma das barras. A dedução destas expressões pode ser encontrada no apêndice A deste trabalho, baseado no qual montamos o quadro 3:

Salientamos ainda que estas forças serão representadas através do vetor de forças de engastamento perfeito local, denotado $\{f\}$, tal qual demonstra a equação 2.8:

$$\{f\} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{Bmatrix} = \begin{Bmatrix} H_0 \\ V_0 \\ M_0 \\ H_1 \\ V_1 \\ M_1 \end{Bmatrix} \tag{2.8}$$

Quadro 3 – Forças de engastamento perfeito para as barras

MT	 0	 1	 2	 3
H_0	$-\frac{1}{6}\left[2q_{x0}+q_{x1}\right]L$			
H_1	$-\frac{1}{6}\left[q_{x0}+2q_{x1}\right]L$			
V_0	$-\frac{1}{20}\left(7q_{y0}+3q_{y1}\right)L$	$-\frac{1}{120}\left(33q_{y0}+12q_{y1}\right)L$	$-\frac{1}{120}\left(48q_{y0}+27q_{y1}\right)L$	$-\frac{1}{6}\left[2q_{y0}+q_{y1}\right]L$
V_1	$-\frac{1}{20}\left(3q_{y0}+7q_{y1}\right)L$	$-\frac{1}{120}\left(27q_{y0}+48q_{y1}\right)L$	$-\frac{1}{120}\left(12q_{y0}+33q_{y1}\right)L$	$-\frac{1}{6}\left[q_{y0}+2q_{y1}\right]L$
M_0	$-\frac{1}{120}\left(6q_{y0}+4q_{y1}\right)$	0	$-\frac{1}{120}\left(8q_{y0}+7q_{y1}\right)L^2$	0
M_1	$\frac{1}{120}\left(4q_{y0}+6q_{y1}\right)$	$\frac{1}{120}\left(7q_{y0}+8q_{y1}\right)L^2$	0	0

Fonte : autor.

2.3.2 Matriz de rigidez local

Para determinar a matriz de rigidez de uma barra pertencente à estrutura, nos utilizaremos das mesmas condições impostas na seção anterior: as barras deverão ser homogêneas e prismáticas, além de estarem submetidas a uma das quatro condições de articulação definidas neste trabalho. Os coeficientes locais, aqui denotados κ_{ij} serão determinados conforme o estabelecido na seção 2.2.2.2 deste trabalho: uma deslocabilidade unitária é imposta na barra, sendo os coeficientes de rigidez computados com base nele.

Neste tópico, Kassimali (2012) desenvolve as expressões para todos as quatro configurações de articulação (Equação 2.9).

2.3.3 Vetor de Ações Nodais

Tal qual explicado na seção 2.3.1, as cargas pontuais da estrutura serão aplicadas exclusivamente em nós, e nunca no meio das barras. Sabendo disto, é possível desmembrar o vetor de forças de engastamento perfeito para criar o vetor de ações nodais.

Este processo tem duas vantagens principais do ponto de vista computacional. Em primeiro lugar, como explicado anteriormente, os casos de carregamento se limitam a cargas linearmente distribuídas, cujos esforços podem ser facilmente deduzidos, como mostrado no apêndice A. Se houvessem cargas aplicadas nas barras, dever-se-ia considerar cada uma delas, calcular cada um de seus esforços separadamente e depois combiná-los para obter o esforço final, sendo este então repassado ao sistema global de coordenadas

pelas regras de correspondência.

$$
(MT=0)\ \ [\kappa]=\begin{bmatrix}
\frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0\\
0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2}\\
0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L}\\
-\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0\\
0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2}\\
0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L}
\end{bmatrix}
$$

$$
(MT=1)\ \ [\kappa]=\begin{bmatrix}
\frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0\\
0 & \frac{3EI}{L^3} & 0 & 0 & -\frac{3EI}{L^3} & \frac{3EI}{L^2}\\
0 & 0 & 0 & 0 & 0 & 0\\
-\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0\\
0 & -\frac{3EI}{L^3} & 0 & 0 & \frac{3EI}{L^3} & -\frac{3EI}{L^2}\\
0 & \frac{3EI}{L^2} & 0 & 0 & -\frac{3EI}{L^2} & -\frac{3EI}{L}
\end{bmatrix}
\tag{2.9}
$$

$$
(MT=2)\ \ [\kappa]=\begin{bmatrix}
\frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0\\
0 & \frac{3EI}{L^3} & \frac{3EI}{L^2} & 0 & -\frac{3EI}{L^3} & 0\\
0 & \frac{3EI}{L^2} & \frac{3EI}{L} & 0 & -\frac{3EI}{L^2} & 0\\
-\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0\\
0 & -\frac{3EI}{L^3} & -\frac{3EI}{L^2} & 0 & \frac{3EI}{L^3} & 0\\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
(MT=3)\ \ [\kappa]=\begin{bmatrix}
\frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0\\
-\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Aí consta a segunda vantagem: basta determinar a que deslocabilidade da estrutura não restringida a carga atuante está associada e montar o vetor de ações nodais $\{A\}$

diretamente no sistema global, de acordo com seus respectivos índices:

$$\{A\} = \begin{Bmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_{n-2} \\ A_{n-1} \\ A_n \end{Bmatrix} \tag{2.10}$$

2.3.4 Vetor de Reações de Apoio

Convém também prever um vetor que conterá o valor das reações de apoio da estrutura restringida. Este vetor, denotado $\{R\}$, conterá um elemento referente a cada uma das deslocabilidades da estrutura, sendo que àqueles que não possuem restrição de deslocamento será atribuído o valor de zero.

2.3.5 Relações entre o sistema de coordenadas global e o local

Como previsto, uma vez que estabelecemos dois sistemas de coordenadas distintos para implementação do método de cálculo, será necessário que um grupo de equações que expressem as relações entre os dois seja estabelecido. Estas relações deverão abarcar todos os aspectos do posicionamento das barras na estrutura, fazendo a correspondência entre os nós da barra no sistema local e os nós da estrutura no sistema global.

2.3.5.1 Matriz de Rotação

No caso de estruturas porticadas, tratadas neste trabalho, é comum que haja elementos orientados em várias direções. Nestes casos, Kassimali (2012) declara que:

> "[...] se torna necessário transformar as relações de rigidez dos membros do pórtico dos seus sistemas de coordenadas locais para o sistema de coordenadas global antes de que possam ser combinadas para estabelecer as relações de rigidez da estrutura como um todo." (KASSIMALI, 2012, pg. 76, tradução nossa)

Isto é, deve-se considerar a rotação dos elementos em relação ao sistema de coordenadas globais para garantir que as suas parcelas de contribuição na rigidez, e também suas forças de engastamento perfeito, sejam consideradas nos eixos apropriados.

Desta forma, Martha (2010) prevê a utilização de uma matriz de rotação $[T_\theta]$ para fazer as transformações entre os sistemas de coordenadas (Equação 2.11). Esta matriz dependerá do ângulo θ da barra em relação ao eixo x da estrutura, isto é, global.

$$[T_\theta] = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\theta & \sin\theta & 0 \\ 0 & 0 & 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Kassimali (2012) ainda estabelece as equações para fazer estas transformações, de modo que se obtenha no sistema de coordenadas global, para cada uma das barras, o seu vetor de engastamento perfeito $\{f_g\}$ e a sua matriz de rigidez $[\kappa_g]$

$$\{f_g\} = [T_\theta]^t\{f\} \tag{2.12}$$

$$[\kappa_g] = [T_\theta]^t[\kappa][T] \tag{2.13}$$

2.3.5.2 Regra da Correspondência

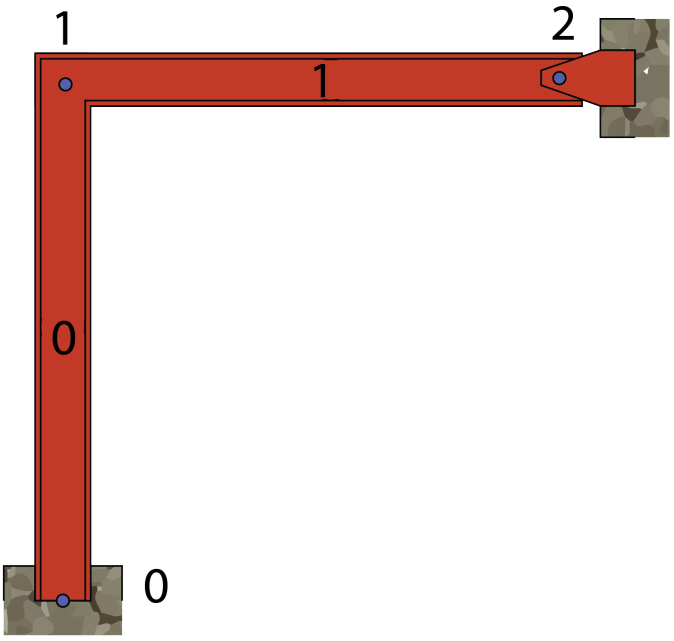
Uma vez determinadas as variáveis da barra em termos das coordenadas globais, podemos começar a combinar os coeficientes de cada elemento na matriz de rigidez global. Para que isto seja possível, deveremos determinar o arranjo dos coeficientes de rigidez e das forças de engastamento perfeito nas suas matrizes e vetores correspondentes, sendo necessário então estabelecer um sistema de numeração para os elementos da estrutura.

Assim, fica determinado que cada nó da estrutura possuirá um índice numérico, que será crescente, contando a partir de 0. O mesmo será feito para as barras, sendo que estas, além de seu índice próprio, também deverão ser identificadas pelo índice dos seus nós. A ordem deles indicará a orientação da barra e, portanto o seu ângulo em relação ao eixo horizontal. A título de ilustração, podemos utilizar a figura 10 como base:

Podemos identificar que a estrutura possui nós de 0 a 2, além de duas barras de índices 0 e 1 que podem ser definidas como: $B_0 : N_0 \to N_1$ e $B_1 : N_1 \to N_2$.

Tendo definido este sistema de indexação, podemos estabelecer as correspondências. Considerando a notação utilizada na equação 2.4 e que cada nó de pórtico tem três

Figura 10 – Pórtico com indicação de seus índices



Fonte: autor.

deslocabilidades, podemos escrever, para o vetor de forças de engastamento perfeito, que:

$$\begin{matrix} \\ N_1 \\ \\ \\ \\ \\ N_n \end{matrix} \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{Bmatrix} \tag{2.14}$$

Isto é, a cada três linhas do vetor, podemos contar um nó, sendo que o vetor contará com uma dimensão de $3n$, sendo n o número de nós da estrutura. Assim, sabendo o índice dos nós de cada uma das barras, é possível saber a qual grau de liberdade da estrutura $f_i g$ será somada. Neste sentido, podemos estabelecer as seguintes relações entre f_i e F_i:

Quadro 4 – Regra da correspondência para pórticos planos

	n
x	$3n$
y	$3n+1$
z	$3n+2$

Sendo assim, para forças agindo no caso do pórtico da figura 10, podemos estabelecer o seguinte vetor F, levando em conta a rotação da barra B_0:

$$\{f_{B_0,g}\} = \begin{Bmatrix} -f_{1,B_0} \\ f_{0,B_0} \\ f_{2,B_0} \\ -f_{4,B_0} \\ f_{3,B_0} \\ f_{5,B_0} \end{Bmatrix}; \{f_{B_1}\} = \{f_{B_1,g}\} = \begin{Bmatrix} f_{0,B_1} \\ f_{1,B_1} \\ f_{2,B_1} \\ f_{3,B_1} \\ f_{4,B_1} \\ f_{5,B_1} \end{Bmatrix}; \{f_{B_1}\} = \begin{Bmatrix} -f_{1,B_0} \\ f_{0,B_0} \\ f_{2,B_0} \\ -f_{4,B_0} + f_{0,B_1} \\ f_{3,B_0} + f_{1,B_1} \\ f_{5,B_0} + f_{2,B_1} \\ f_{3,B_1} \\ f_{4,B_1} \\ f_{5,B_1} \end{Bmatrix}$$

Esta mesma lógica pode ser seguida para a matriz de rigidez da estrutura, sendo que esta, por apresentar uma organização bidimensional, segue a seguinte configuração:

$$\begin{array}{c} \\ N_1 \\ \\ \\ \\ \\ \\ N_n \end{array} \begin{array}{c} \begin{array}{ccccccc} N_1 & & & & & N_n & \end{array} \\ \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & & k_{0,n-2} & k_{0,n-1} & k_{0,n} \\ k_{1,0} & k_{1,1} & k_{1,2} & \dots & k_{1,n-2} & k_{1,n-1} & k_{1,n} \\ k_{2,0} & k_{2,1} & k_{2,2} & & k_{2,n-2} & k_{2,n-1} & k_{2,n} \\ & \vdots & & \ddots & & \vdots & \\ k_{n-2,0} & k_{n-2,1} & k_{n-2,2} & & k_{0,n-2} & k_{0,n-1} & k_{0,n} \\ k_{n-1,0} & k_{n-1,1} & k_{n-1,2} & \dots & k_{1,n-2} & k_{1,n-1} & k_{1,n} \\ k_{n,0} & k_{n-,1} & k_{n,2} & & k_{2,n-2} & k_{2,n-1} & k_{2,n} \end{bmatrix} \end{array}$$

Assim, teremos, no caso da matriz de rigidez, a relação expressa no quadro 4:

Quadro 5 – Regra da correspondência para pórticos planos

	N_0	N_1
x	$3n_0$	$3n_1$
y	$3n_0 + 1$	$3n_1 + 1$
z	$3n_0 + 2$	$3n_1 + 2$

2.3.6 Resolução do Sistema

Ao final da composição da equação de compatibilidade da estrutura, com os elementos que foram adicionados ao longo desta seção, teremos o representado na equação a seguir.

Tendo a equação da estrutura não restringida, poderemos aplicar as condições de contorno relativas às restrições de deslocamento. Para isto, igualaremos todas as deslocabilidades restringidas a zero, o que nos levará a definir um sistema de equações reduzido. Para o caso do pórtico da figura 10, temos representada a equação da estrutura restringida abaixo:

$$\begin{Bmatrix} R_0 \\ R_1 \\ R_2 \\ 0 \\ 0 \\ 0 \\ R_6 \\ R_7 \\ 0 \end{Bmatrix} = \begin{Bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \end{Bmatrix} + \begin{Bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \end{Bmatrix} + \begin{bmatrix} K_{0,0} & K_{0,1} & K_{0,2} & K_{0,3} & K_{0,4} & K_{0,5} & K_{0,6} & K_{0,7} & K_{0,8} \\ K_{1,0} & K_{1,1} & K_{1,2} & K_{1,3} & K_{1,4} & K_{1,5} & K_{1,6} & K_{1,7} & K_{1,8} \\ K_{2,0} & K_{2,1} & K_{2,2} & K_{2,3} & K_{2,4} & K_{2,5} & K_{2,6} & K_{2,7} & K_{2,8} \\ K_{3,0} & K_{3,1} & K_{3,2} & K_{3,3} & K_{3,4} & K_{3,5} & K_{3,6} & K_{3,7} & K_{3,8} \\ K_{4,0} & K_{4,1} & K_{4,2} & K_{4,3} & K_{4,4} & K_{4,5} & K_{4,6} & K_{4,7} & K_{4,8} \\ K_{5,0} & K_{5,1} & K_{5,2} & K_{5,3} & K_{5,4} & K_{5,5} & K_{5,6} & K_{5,7} & K_{5,8} \\ K_{6,0} & K_{6,1} & K_{6,2} & K_{6,3} & K_{6,4} & K_{6,5} & K_{6,6} & K_{6,7} & K_{6,8} \\ K_{7,0} & K_{7,1} & K_{7,2} & K_{7,3} & K_{7,4} & K_{7,5} & K_{7,6} & K_{7,7} & K_{7,8} \\ K_{8,0} & K_{8,1} & K_{8,2} & K_{8,3} & K_{8,4} & K_{8,5} & K_{8,6} & K_{8,7} & K_{8,8} \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ D_3 \\ D_4 \\ D_5 \\ 0 \\ 0 \\ D_8 \end{Bmatrix}$$

Vendo que quatro das equações podem ser descartadas neste primeiro momento, pois sabemos que os deslocamentos associados a elas serão iguais a zero (deslocabilidades restringidas), podemos obter o sistema reduzido, para o qual determinaremos os valores dos D_i não nulos:

$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} A_3 \\ A_4 \\ A_5 \\ A_8 \end{Bmatrix} + \begin{Bmatrix} F_3 \\ F_4 \\ F_5 \\ F_8 \end{Bmatrix} + \begin{bmatrix} K_{3,3} & K_{3,4} & K_{3,5} & K_{3,8} \\ K_{4,3} & K_{4,4} & K_{4,5} & K_{4,8} \\ K_{5,3} & K_{5,4} & K_{5,5} & K_{5,8} \\ K_{8,3} & K_{8,4} & K_{8,5} & K_{8,8} \end{bmatrix} \begin{Bmatrix} D_3 \\ D_4 \\ D_5 \\ D_8 \end{Bmatrix}$$

Uma vez determinados os valores dos deslocamentos, será possível utilizar as equações complementares para determinar os valores das reações, de modo que teremos:

$$\begin{Bmatrix} R_0 \\ R_1 \\ R_2 \\ R_6 \\ R_7 \end{Bmatrix} = \begin{Bmatrix} A_0 \\ A_1 \\ A_2 \\ A_6 \\ A_7 \end{Bmatrix} + \begin{Bmatrix} F_0 \\ F_1 \\ F_2 \\ F_6 \\ F_7 \end{Bmatrix} + \begin{bmatrix} K_{0,3} & K_{0,4} & K_{0,5} & K_{0,8} \\ K_{1,3} & K_{1,4} & K_{1,5} & K_{1,8} \\ K_{2,3} & K_{2,4} & K_{2,5} & K_{2,8} \\ K_{6,3} & K_{6,4} & K_{6,5} & K_{6,8} \\ K_{7,3} & K_{7,4} & K_{7,5} & K_{7,8} \end{bmatrix} \begin{Bmatrix} D_3 \\ D_4 \\ D_5 \\ D_8 \end{Bmatrix}$$

2.3.7 Diagramas da estrutura

Para a determinação dos diagrama da estrutura será necessário, em um primeiro momento, determinar quais os esforços internos de cada barra nos seus respectivos sistemas de coordenadas locais. Isto pode ser alcançado realizando o processo inverso ao feito para

compor os vetores e matriz global. Assim, poderemos definir $\{f\}$,$\{d\}$ e $\{k\}$, de modo que será possível resolver as equações de compatibilidade locais (Equação 2.15), determinando o valor das reações locais $\{r\}$ nos apoios de cada uma das barras. Deste modo, Kassimali (2012) estata que:

$$\{r\} = \{f\} + [k]\{d\} \tag{2.15}$$

Será necessário então determinar $\{d\}$, visto que os demais componentes já foram calculados previamente. Para isto, podemos nos utilizar da regra da correspondência para determinar as componentes do vetor deslocamento da barra no sistema global $\{d_g\}$ e, pela equação 2.16 projetá-las no sistema local.

$$\{d\} = [T_\theta]\{d_g\} \tag{2.16}$$

Tendo então os valores de $\{r\}$, descobrimos as reações dos esforços nas barras, que coincidem com o valor dos esforços internos nestes pontos. Logo, podemos nos valer do que diz Sussekind (1981) e estabelecer a equação dos diagramas de esforços internos em cada uma das barras. O desenvolvimento destas equações para barras prismáticas e cargas distribuídas linearmente variáveis pode ser encontrada no apêndice B.

Já para o diagrama de deformações da estrutura, nos utilizaremos de um processo que diverge do utilizado anteriormente. Segundo McGuire et al. (2000):

> "A configuração deformada de elementos estruturais mais simples, como os estudados previamente, pode ser encontrada resolvendo as equações diferencias que governam o comportamento destes elementos." (MCGUIRE et al., 2000, pg. 175, tradução nossa)

Assim, podemos utilizar as equações 2.17 abaixo para determinar os deslocamentos no eixo x, y e z:

$$\frac{\mathrm{d}^4 v}{\mathrm{d}x^4} = \frac{1}{EI} q_y(x); \quad \frac{du}{dx} = \frac{1}{EA} q_x(x) \tag{2.17}$$

Contudo, McGuire et al. (2000) afirma que estas equações poderão ser aproximadas com boa precisão através de uma expressão algébrica em termos dos deslocamento nas extremidades da barra:

$$d(x) = \sum_{i=0}^{n} \mathbb{C}_i d_i$$

, onde $\mathbb{C}_i$ assume o papel de coeficiente de forma da curva, d_i dos deslocamentos no nó e n o número de graus de liberdade do elemento em questão. O desenvolvimento destas expressões poderá ser encontrado na seção B.4 do respectivo apêndice.

2.4 Eliminação Gaussiana para resolução de sistemas lineares

Um dos pontos essenciais do método analisado é a resolução do sistema de equações lineares para precisar o valor dos deslocamentos da estrutura. Neste sentido, se torna imprescindível estabelecer um método de resolução cuja implantação seja simples e sistemática. Por ser um método de operações simples, a eliminação gaussiana acaba tornando-se um candidato forte para esta parte do processo. Nas palavras de Atkinson (1989):

> "Este [eliminação gaussiana] é o nome formal dado ao método de solucionar sistemas de equações lineares através da eliminação sucessiva de incógnitas e redução do sistema a uma menor ordem." (ATKINSON, 1989, pg. 508, tradução nossa)

Assim, para resolver um dado sistema $[A]\{x\} = \{b\}$, deveremos reduzi-lo a um sistema equivalente $[I]\{x\} = \{g\}$, onde $[I]$ representa a matriz identidade de dimensões iguais às de $[A]$. Para fazer isto, nos valeremos de uma representação particular do sistema de equações, a qual denominaremos matriz aumentada (Equação 2.18). Esta matriz é formada pela composição de $[A]$ e $\{b\}$, levando os mesmos coeficientes da matriz $[A]$, mas à sua última coluna acoplada o vetor de resultados $\{b\}$.

$$[A \mid b\,] = \left[\begin{array}{cccc|c} A_{00} & A_{01} & & A_{0n} & b_0 \\ A_{10} & A_{11} & \cdots & A_{1n} & b_1 \\ \vdots & & \ddots & \vdots & \vdots \\ A_{n0} & A_{n1} & \ldots & A_{nn} & b_n \end{array}\right] \tag{2.18}$$

Sobre esta matriz poderão ser realizadas três operações principais expostas por Atkinson (1989) cujo objetivo é reduzir $[A]$ para $[I]$:

$L_i \Leftarrow L_i + L_j$: Multiplicação de uma linha L_i por uma constante $c \neq 0$;

$L_i \Leftarrow L_i + L_j$: Soma de duas linhas L_i e L_j da matriz;

$L_i \Leftrightarrow L_j$: Troca de de duas linhas L_i e L_j na matriz.

Logo, podemos estabelecer uma rotina para a resolução dos sistemas lineares presentes no método. Tal rotina também é exposta por Atkinson (1989), que prevê o uso de recursão para implementá-la. Neste caso, definiremos um índice $0 \leq i < n$ que definirá

qual linha está sendo estudada – n sendo o número de linhas da matriz aumentada. Para cada linha L_i o processo será repetido, de forma que se tenha operado sobre todas as linhas da matriz ao fim do algoritmo.

1. $i = 0$;
2. Se $a_{ii} \neq 0, \quad L_i \Leftarrow L_i \times \frac{1}{a_i i}$; senão $L_i \Leftrightarrow L_j$, onde L_j é a primeira linha que contém o elemento $a_{ji} \neq 0$. Se $i = n - 1$ ou $j = n$, a matriz é singular e portanto apresenta infinitas soluções;
3. $L_j \Leftarrow L_j - a_{ji} L_j$, com $0 \leq j < n \ \mid \ j \neq i$;
4. Se $i < n, i = i + 1$ e retorna para o passo 2; senão sai do algoritmo;

Por fim, a matriz aumentada do sistema, agora equivalente, assumirá a forma da equação 2.19, donde facilmente se extrai o valor de cada uma das incógnitas

$$[I \mid g\,] = \left[\begin{array}{cccc|c} 1 & 0 & & 0 & g_0 \\ 0 & 1 & \cdots & 0 & g_1 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & g_n \end{array}\right] \tag{2.19}$$

2.5 Ferramentas Computacionais

Para dar procedência a este trabalho, é importante que se compreenda como funcionam as ferramentas que serão utilizadas para compor o aplicativo. Por se tratar do desenvolvimento de um produto, ha um aspecto holístico na escolha destas ferramentas, devendo-se ter em mente que este processo não se limita à pura programação, se estendendo também à confecção da interface programa-usuário, à criação dos gráficos utilizados e também à estruturação do aplicativo como um todo. Todavia, por tratar-se de um trabalho acadêmico voltado aos aspectos estruturais do desenvolvimento, nos ateremos a este âmbito, sem dissertar sobre os demais pontos do processo.

2.5.1 Modo de programação

O desenvolvimento em *Android* apresenta um ponto divergente do trivial: nos deparamos com uma plataforma estratificada, que se utiliza de duas linguagens de programação para a confecção dos seus aplicativos. Nas palavras de Jackson (2011, pg. 41, tradução nossa), "Tudo no ambiente de desenvolvimento para *Android*, inclusive os aplicativos nele incluídos podem ser programados com uma combinação de *Java* e *XML* [...]".

Esta estratificação visa economizar ao usuário várias linhas de código, já que a programação em *XML* para *Android* conta com toda uma biblioteca de funções preprogramadas, restando apenas a definição de propriedades inerentes a cada uma destas funções. Em linhas gerais, podemos dizer que a linguagem *Java* se ocupa do funcionamento da aplicação, enquanto o *XML* se ocupa das características de leiaute (JACKSON, 2011).

Neste sentido, de modo que seja possível justificar as escolhas de estruturação adotadas neste trabalho, é imprescindível que seja fornecido um breve panorama de cada uma das linguagens utilizadas.

2.5.1.1 *Java SE*

O ponto que mais se destaca na linguagem Java, no que concerne este trabalho, é o fato de ser orientada a objeto. Isto quer dizer que toda a estruturação do código será criada em torno da definição dos objetos inerentes ao programa e também como se comportam perante certas situações, sendo para isto utilizadas classes e métodos, respectivamente (BRYANT, 2012). Podemos contextualizar esta definição para o caso das matrizes. Segundo o que foi dito, pode-se considerar a classe **Matriz**, cujas variáveis englobarão as suas dimensões m e n e também os seus coeficientes a_{ij}. Já para os métodos, podemos exemplificá-los com: multiplicação da matriz por um coeficiente, cálculo do seu determinante e multiplicação de duas matrizes. Esta perspectiva é especialmente interessante para a construção do aplicativo, visto que utilizaremos este conceito de objeto durante todo o seu desenvolvimento. Além do objeto **matriz**, podemos ainda citar os objetos **barra**, **perfil** e **material**. A própria estrutura poderá ser considerada como um, sendo constituída por um série de barras, nós, materiais, etc.

Ao encontro do que foi dito anteriormente, pormenores da linguagem como a sua sintaxe não serão abordados neste trabalho. Alternativamente, optou-se por representar as rotinas constituintes do programa através de fluxogramas, de modo que a visualização do funcionamento de cada uma das classes, e de seus respectivos métodos, seja facilitado para o leitor.

2.5.1.2 XML

A linguagem *XML*, no que se refere ao desenvolvimento para Android, carrega um grande trunfo: apesar de não ser tecnicamente uma linguagem de programação, se utiliza de *tags* para definir aspectos básicos de leiaute e interface programa-usuário (JACKSON, 2011). Estas *tags* denotam funções préprogramadas como botões, imagens e menus, para os quais deverão ser definidos uma série de parâmetros, como altura, largura e posicionamento.

Estes elementos poderão ser posteriormente customizados para que atendam às necessidades do programa, como um botão que executa uma determinada ação. Um exemplo de *tag* em *XML* pode ser vista na figura 11.

Figura 11 – Exemplo de tag XML para Android

```
<ImageButton
    android:id="@+id/btn_menu"
    android:layout_width="64dp"
    android:layout_height="64dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="5dp"
    android:layout_marginBottom="5dp"
    android:background="@drawable/cog"
    />
```

Fonte: autor.

Na figura mostrada acima, percebemos que se trata de um botão de imagem, com dimensões 64×64, que se alinha com o canto inferior direito da tela, afastando 5 unidades de medida de margem e que é representado pela figura *cog.png*. Também vemos que foi com poucas linhas de código que tal elemento foi definido, não havendo a necessidade de se apelar para recursos programáticos mais complexos.

3 Metodologia

O trabalho foi realizado em três grandes momentos, cujos aspectos limítrofes tem origem no cunho dos procedimentos realizados em cada um deles. Esta divisão foi realizada para que se assegurasse uma ordem cronológica bem estabelecida, sem que houvesse necessidade de revisitar pontos que já haviam sido dados como concluídos. Destarte, é cabível que estes três momentos sejam explicados neste capítulo.

3.1 Revisão Bibliográfica e Estruturação do Aplicativo

Apesar de apresentarem procedimentos divergentes, a revisão bibliográfica do método de cálculo e a concomitante estruturação do aplicativo convergem no sentido de prepararem um embasamento sobre o qual o produto final será apoiado. Assim, neste primeiro momento, buscou-se reunir o máximo de informações possível sobre o método de cálculos e também sobre todas as ferramentas que este viria a utilizar. Tendo em mãos estas informações, foi possível definir a quais necessidades o aplicativo deveria atender, bem como qual deveria ser a sua estrutura gráfica e matemática. Em contrapartida, também buscou-se entender quais eram as limitações da plataforma e dos dispositivos, de modo que o método de cálculo também pudesse ser adaptado a estas.

3.2 Programação do Aplicativo

Tendo definida toda a parte teórica, deu-se início à parte prática do trabalho, donde resultou toda a codificação utilizada no aplicativo. Os códigos foram todos realizados de forma a atender as necessidades matemáticas e gráficas estabelecidas no primeiro momento do trabalho, sendo que buscou-se deixá-los o mais eficiente e eficaz possível. Também procurou-se ajuda no que diz respeito à estruturação do código e modelos de códigos em fóruns online de programação, além de ter-se disponibilizado trechos de código, rotinas e sub-rotinas de cálculo para que estas fossem avaliadas pelos frequentadores de tais fóruns.

3.3 Conferência de Resultados

No terceiro momento, deu-se a validação dos resultados apresentados pelo aplicativo. Por se tratar de um processo iterativo, é interessante que o desenvolvimento do aplicativo, sobretudo o que concerne a rotina de cálculo, seja norteado por resultados comprovadamente corretos. Neste intuito buscou-se fazer uma compilação de estruturas já resolvidas, com

as quais poderiam ser analisados todos os elementos do problema, desde as matrizes de rigidez locais até as reações nos apoios da estrutura.

Assim, nos baseamos nos exemplos resolvidos de Martha (2010), Sussekind (1981), Kassimali (2012), Ghali et al. (2009) e McGuire et al. (2000). A partir deles, foi possível calibrar os resultados do programa, de modo que se encontrassem potenciais erros de programação ou problemas de precisão. Buscou-se também abranger situações diversas, variando vinculações, cargas e condições de apoio.

Uma vez corrigidas todas as inconsistências, lançamos mão de programas de análise estrutural plana para que fosse possível verificar os cálculos com um número ainda maior e mais variado de estruturas.

4 Estruturação do Aplicativo

O primeiro passo para a codificação do aplicativo é a definição da sua estrutura analítica, isto é, quais serão seus componentes e como interagirão entre si. Esta etapa é de extrema importância, pois molda o cerne do código e define a perspectiva sob a qual serão concebidos os seus aspectos subsequentes. Se tratando da linguagem Java, ela vem quase que intuitivamente, pois será idealizada de forma a parametrizar os objetos componentes.

Outra consideração importante que deve ser feita é o fato de que estes objetos não se limitam aos constituintes físicos da estrutura, mas também a ferramentas de apoio matemático e elementos de leiaute não contemplados pelas *tags XML*. Todos os três grupos são utilizados para a composição do produto final desta dissertação, mas nos ateremos aos dois primeiros, dando enfoque aos aspectos mais essenciais, vis à vis da grande quantidade de classes e objetos que podem vir a ser necessários para a confecção de um programa. Logo, elementos cujos propósitos estejam ligados à interface usuário-programa não serão tratados neste trabalho, o mesmo valendo para funções matemáticas simples, cujo funcionamento pode ser prontamente deduzido a partir dos códigos anexos.

No mesmo sentido, alguns detalhes sobre os métodos ainda devem ser levantados, visto que há os que serão comum a todas as classes. Exemplificando sob um ponto de vista prático, há o fato de que, para que sejam usados ao longo do código, estes objetos deverão ser declarados, de sorte que será necessário um método que faça isto, denotado construtor. Este poderá tomar como entrada todas as variáveis que compõem o objeto em questão, ou então apenas uma parte delas, deixando as demais para o processamento posterior. De outra perspectiva, temos também os métodos que atribuirão valores às variáveis dos objetos e aqueles que os obterão, os quais são trivialmente denominados de *setters* e *getters*, respectivamente. Estes dois últimos tipos de método serão um lugar comum na programação do aplicativo e, tendo uma natureza pouco complexa, não é interessante quedar-se em sua listagem: via de regra será admitido que foram previstos métodos que realizem estas operações para cada uma das variáveis.

Neste trabalho representaremos as rotinas referentes aos métodos pertinentes, ou seja, aqueles que são essenciais à análise da estrutura, através de fluxogramas. Estas seguirão a convenção estabelecida na norma ISO 5807:1985 (ISO. . . ,).

No tocante aos demais métodos, serão representados em formato de quadro, acompanhados de uma pequena documentação sobre o seu propósito e de seus dados de entrada e saída (retorno). Cabe também destacar que os códigos referentes aos elementos aqui descritos podem ser encontrados no Apêndice D.

4.1 Ferramentas de Apoio Matemático

Para a confecção dos itens propostos no primeiro capítulo deste trabalho, identificamos duas necessidades no que concerne a instrumentação matemática: a criação de uma classe que defina o comportamento de pontos cartesianos e outra que defina o comportamento de matrizes.

4.1.1 Classe Ponto

A primeira delas, uma classe de pontos, surge do fato de que a entrada de dados do programa será baseada majoritariamente na inserção de coordenadas em um plano cartesiano. Para que isto seja possível, é preciso um modelo robusto que consiga lidar com os dados de entrada e que ofereça em troca todos os recursos inerentes a um ambiente cartesiano. Isto quer dizer que a implantação de funções como distância entre pontos, distância entre ponto e reta e soma de pontos serão indispensáveis para o processamento dos dados fornecidos pelo usuário através da interface.

Por definição, um ponto é uma entidade geométrica adimensional cuja posição espacial pode ser representada através de coordenadas. A quantidade de coordenadas utilizadas dependerá do espaço em qual está inserido – no que concerne o assunto estudado, este espaço se restringe a um plano, o qual demanda apenas duas coordenadas. Assim, um ponto P poderá ser representado por $P = P(x, y)$, isto é, atribuiremos a este objeto as variáveis x e y.

Dois construtores serão contabilizados para a classe ponto: o primeiro receberá tanto a coordenada x quanto a y do objeto, enquanto o segundo não possuirá dados de entrada, criando um ponto $P(0, 0)$ locado na origem cartesiana.

A partir deste conceito, podemos listar os métodos no quadro 6. No que diz respeito às equações utilizadas em cada um deles, cabe informar que podem ser verificadas no próprio código da classe, sendo que foram adaptadas de Steinbruch e Winterle (1987a)

4.1.2 Classe Matriz

Passamos então à classe que definirá as matrizes utilizadas ao longo da análise da estrutura. No decorrer deste trabalho foram abundantes os exemplos de quão essenciais as matrizes serão para a etapa de análise estrutural. Em vista disto, é imprescindível que elas sejam caracterizadas pelo menor conjunto de elementos possível e que seus métodos sejam eficientes: uma construção grosseira desta classe pode levar a sérios problemas de performance.

Do senso comum, indicamos três elementos principais que descrevem uma matriz: as suas dimensões, aí contabilizando o número de linhas m e de colunas n, e uma tabela

Quadro 6 – Tipo do membro em relação a sua rotulação

	Descrição	Entrada	Saída
getDistancia	Calcula a distância entre dois pontos.	**Ponto** $A(x_A, y_A)$, **Ponto** $B(x_B, y_B)$	**Número** d
getAngulo	Calcula o ângulo de uma reta formada por dois pontos em relação com a horizontal, sendo o antihorário positivo.	**Ponto** $A(x_A, y_A)$, **Ponto** $B(x_B, y_B)$	**Número** θ
isEqual	Compara dois pontos e verifica se suas coordenadas são iguais.	**Ponto** $A(x_A, y_A)$, **Ponto** $B(x_B, y_B)$	**Booleana** $true, false$
mult	Multiplica o ponto por uma constante.	**Ponto** $A(x_A, y_A)$, **Número** c	**Ponto** $B(c \times x_A, c \times y_A)$
toString	Retorna uma representação do ponto em formato de texto.	**Ponto** $A(x_A, y_A)$	**Texto** ponto-Texto
coinc	Verifica se duas semirretas, formadas por dois pontos cada uma, são coincidentes.	**Ponto** $A_1(x_{A1}, y_{A1})$, **Ponto** $A_2(x_{A2}, y_{A2})$, **Ponto** $B_1(x_{B1}, y_{B1})$, **Ponto** $B_2(x_{B2}, y_{B2})$	**Booleana** $true, false$
distReta	Mede a menor distância entre um ponto P até uma reta formada por dois pontos.	**Ponto** $P(x_P, y_P)$, **Ponto** $A_1(x_{A1}, y_{A1})$, **Ponto** $A_2(x_{A2}, y_{A2})$	**Número** distância
cruzamento	Identifica o ponto de cruzamento entre duas retas definidas por dois pontos cada	**Ponto** $A_1(x_{A1}, y_{A1})$, **Ponto** $A_2(x_{A2}, y_{A2})$, **Ponto** $B_1(x_{B1}, y_{B1})$, **Ponto** $B_2(x_{B2}, y_{B2})$	**Ponto** C

Fonte: autor

dados cujas células são os coeficientes. No caso da tabela *dados*, utilizaremos um vetor bidimensional para indexar os coeficientes nela contidos, bem como é feito usualmente para matrizes. Assim, trabalharemos frequentemente com índices i,j e k, o que todavia não implica no fato de eles serem tomados como variáveis do objetos, visto que não são fatores caracterizantes.

No caso das matrizes, poderemos definir três construtores para estes elementos. O mais evidente é a construção de um objeto matriz a partir de uma tabela *dados* fornecida, o que implica em que a matriz terá dimensões condizentes com as da tabela. Desta noção vem o segundo construtor, que definirá uma matriz de coeficientes nulos baseado em duas dimensões $m \times n$. Por fim, definiremos também um construtor de matrizes quadradas que, como o anterior, confeccionará uma matriz $m \times m$ de coeficientes nulos, sendo que a sua entrada será o único número inteiro m.

A estruturação teórica dos métodos das matrizes foi feita baseada em Steinbruch e Winterle (1987b), com exceção da resolução do sistema de equações, cuja idealização foi feita a partir dos dizeres de Atkinson (1989) na seção 2.4. Deste modo, monta-se o quadro 7. No caso do método *resolve*, é interessante ainda que seja mostrado o fluxograma de sua rotina (Figura 12), vis à vis de sua importância no processo de cálculo.

4.2 Objetos constituintes da estrutura

Tendo definido as ferramentas de apoio matemático, podemos passar às classes que regerão o comportamento da estrutura como um objeto físico. Para isto, podemos nos valer de uma formação de grupo, ou seja, vamos organizá-los hierarquicamente. A título de exemplo, podemos estabelecer o seguinte ramo da estrutura: um pórtico possui barras que, por sua vez, possuem materiais que, por sua vez, são caracterizados por um índice, um nome, um módulo de elasticidade e uma densidade. Este tipo de organização nos ajudará a definir além das relações de interdependência, quais pontos deverão ser tratados como objetos e quais deverão ser tratados como simples variáveis.

Analisando a figura 13, percebemos um padrão que pode ser usado para triar as classes das variáveis: de modo geral, serão adotados como variáveis os níveis mais exteriores do esquema, enquanto os intermediários serão considerados como objetos. Outro aspecto desta hierarquia é que alguns elementos são referenciados como variáveis de outros. Um bom exemplo são as barras, que possuem variáveis indicando qual seu material de composição, qual seu perfil e quais os nós que as extremam. Logo, se considerarmos que a grande parte destas variáveis se repetirá, torna-se lógico que estes objetos devem ser criados à parte das barras que os possuem. Assim, preveremos para eles um índice que nos permitirá fazer a relação entre estas classes, de modo que, se quisermos atribuir uma determinada carga pontual a um nó, bastará que o índice *id* desta carga seja atribuído à

Quadro 7 – Tipo do membro em relação a sua rotulação

	Descrição	Entrada	Saída
rigidez	Monta a matriz de rigidez de acordo com os dados da barra (módulo de elasticidade E, área A, momento de inércia I, comprimento L e articulação mt).	**Número** E, **Número** A, **Número** I, **Número** L, **Inteiro** mt	**Matriz** $[R]$
rotação	Monta uma matriz de rotação de uma barra rotacionada θ graus da horizontal	**Número** θ	**Matriz** $[T_\theta]$
engPerfeito	Monta o vetor de engastamento perfeito de uma barra do tipo mt e comprimento L a partir de sua carga distribuída Q	**CDistribuída** Q, **Número** L, **Inteiro** mt	**Matriz** $[f]$
multLinha	Multiplica a linha i da matriz $[A]$ por uma constante c	**Matriz** $[A]$, **Inteiro** i, **Número** c	**Matriz** $[A]$
swapLinha	Troca de posição as linhas i e j da matriz $[A]$	**Matriz** $[A]$, **Inteiro** i, **Inteiro** j	**Matriz** $[A]$
transpoe	Transpoe a matriz $[A]$	**Matriz** $[A]$	**Matriz** $[A]$
mult	Multiplica a matriz $[A]$ pela $[B]$, nesta ordem	**Matriz** $[A]$, **Matriz** $[B]$	**Matriz** $[C]$
somaCelula	Soma um valor c à célula localizada na posição i,j da matriz $[A]$	**Matriz** $[A]$, **Inteiro** i, **Inteiro** j, **Número** c	**Matriz** $[C]$
soma	Soma as matrizes $[A]$ e $[B]$	**Matriz** $[A]$, **Matriz** $[B]$	**Matriz** $[C]$
subtrai	Subtrai as matrizes $[A]$ e $[B]$	**Matriz** $[A]$, **Matriz** $[B]$	**Matriz** $[C]$
linhaSMC	Soma a linha j à linha i multiplicada pela constante c	**Matriz** $[A]$, **Inteiro** i, **Inteiro** $[j]$, **Número** $[c]$	**Matriz** $[A]$
toString	Cria uma representação inteligível da matriz $[A]$ em formato de texto	**Matriz** $[A]$	**Texto** $[t]$
resolve	Resolve o sistema da matriz aumentada $[S]$ e retorna o vetor de resultados $\{R\}$	**Matriz** $[S]$	**Matriz** $\{R\}$

Fonte: autor

variável *cargapontual* no objeto *nó*.

De todo modo, perdura a necessidade de uma definição formal dos métodos e variáveis. Assim, utilizaremos a figura 13 como base, cerceando-a de fora a dentro.

4.2.1 Classe Carga Pontual

Cargas pontuais são carregamentos localizados em um determinado ponto da estrutura, podendo agir em três direções para estruturas planas: uma força no eixo x, uma força no eixo y e um momento fletor no eixo z. Como neste trabalho cada nó possuirá uma única carga pontual, o objeto deverá permitir a inserção simultânea de carregamentos nestas três direções. Logo, identificaremos para esta classe, além do seu índice *id* e do seu *nome*, um vetor unidimensional de números cujos elementos 0,1,2 designarão os valores das cargas P_x, P_y e P_z, respectivamente.

No que diz respeito aos seus construtores, somente um será adotado, sendo que tomará todas as variáveis constituintes da classe: *id*, *nome*, P_x, P_y e P_z.

Já quanto aos métodos utilizados nesta classe, nos limitaremos ao *toString*, que retornará em texto uma representação inteligível da carga pontual.

4.2.2 Carga Distribuída

Como já definido anteriormente, as cargas distribuídas das quais este trabalho se encarregará são bem definidas: agem em ambos os eixos x e y locais da barras, podendo apresentar valores de carregamento diferentes no início e no final da barra. Por possuírem uma variação linear, não será necessário nenhum valor suplementar aos de carregamento inicial e final para definir a sua curva de distribuição. Assim, fica claro que serão necessárias as variáveis q_{x0}, q_{y0}, q_{x1}, q_{y1}, além das recorrentes *id* e *nome*.

Assim como com as cargas pontuais, para as distribuídas será definido um único construtor, que tomará como entrada todos os dados listados acima. Sobre os métodos, somente consideraremos o *toString*, sendo que os demais se resumirão a *setters* e *getters*.

4.2.3 Material

Devido às simplificações feitas no método, a classe Material pode ser definida em poucas variáveis. Já que temos um material isotrópico e homogêneo, nos limitaremos em definir o seu módulo de elasticidade E e também a sua massa específica ρ. No momento, a variável ρ não terá aplicação prática no cálculo, mas poderá ser utilizada futuramente para que cargas de peso próprio sejam calculadas de acordo com a área da seção dos elementos.

Para esta classe também será definido um único construtor levando todas as variáveis e, como métodos, apenas o *toString*.

4.2.4 Perfil

Para a classe Perfil, deveremos levar em consideração que as seções das barras podem ter vários formatos, dos quais precisaremos calcular a área e a inércia. Destarte,

serão definidos os formatos de perfil que poderão ser utilizados no programa e também as equações para calcular suas respectivas áreas e inércias, o que será feito a partir de um conjunto de dimensões d_i informadas pelo usuário:

a) **Genérica**: o usuário insere tanto a área quanto o momento de inércia

b) **Circular**: insere o diâmetro

c) **Retangular**: insere a base e a altura

d) **Circular vazada**:insere o diâmetro e a espessura da casca

e) **Retangular vazada**: insere a base, a altura, a espessura da alma e a espessura da mesa

f) **Tê**: insere a base, a altura, a espessura da alma e a espessura da mesa

g) **I**: insere a base, a altura, a espessura da alma e a espessura da mesa

Tal cálculo dispensa, para casos comuns, a necessidade do usuário de calcular estes valores previamente à análise estrutural. Ainda deve-se perceber que não é possível alterar a ângulo do perfil em relação ao eixo da barra: este se dará de modo que o momento de inércia resistente seja sempre o calculado.

No que concerne o cálculo do momento de inércia, nos valeremos do teorema dos eixos paralelos, exposto por Beer et al. (2010), para decompor as seções em geometria menos complexas, como por exemplo círculos e retângulos. É notável que este método requer que se conheça a posição do centro geométrico da figura, o que poderá ser definido a partir da equação 4.1.

$$\bar{x} = \frac{\sum \bar{x}_i A_i}{\sum A_i}; \qquad \bar{y} = \frac{\sum \bar{y}_i A_i}{\sum A_i} \tag{4.1}$$

Tendo definido estes pontos, já podemos enumerar as variáveis da classe perfil: primeiramente, temos as usuais *id* e *nome*, então as dimensões d_i (quatro para a seção mais geometricamente complexa), a área A, o momento de inércia I e, por fim, o índice *tipo* do perfil.

Nesta classe também só haverá um construtor, porém ao contrário do que tem sido feito até agora, ele não tomará como entrada todas as variáveis constituintes do objeto: para a sua área e momento de inércia serão previstos métodos de cálculo. Desta forma, podemos montar o quadro 8.

Quadro 8 – Métodos da classe perfil

	Descrição	Entrada	Saída
toString	Retorna uma representação inteligível do perfil P em formato de texto *desc*	**Perfil** P	**Texto** desc
calcArea	Retorna a área A do perfil de acordo com as suas dimensões d_i e seu tipo *tipo*	**Número** d_0, **Número** d_1, **Número** d_2, **Número** d_3, **Inteiro** tipo	**Número** A
calcArea	Retorna o momento de inércia I do perfil de acordo com as suas dimensões d_i e seu tipo *tipo*	**Número** d_0, **Número** d_1, **Número** d_2, **Número** d_3, **Inteiro** tipo	**Número** I

Fonte: autor

4.2.5 Classe Apoio

Esta classe corresponde às restrições ao deslocamento adotadas nos nós. Como sabemos, estas restrições podem atuar em três direções, o que nos demanda três variáveis booleanas (verdadeiro/falso) para descrever o tipo de suporte, resultando em seis possíveis combinações. Para que as verificações sejam mais rápidas, sem que seja necessário passar por cada uma das três variáveis sempre que se queira definir o grau de restrição, adotaremos um índice para cada combinação. Assim, restrições em x terão valor de 1, em y valor de 2 e em z valor de 4, sendo que a soma destes três valores resultará na configuração final do apoio. Logo, um apoio restrito no eixo z e x, terá um índice de 5.

Além disso, não serão definidos construtores para esta classe, visto que não haverá objetos propriamente ditos sendo criados. A classe servirá apenas para gerir a ideia de um apoio, apresentando métodos para identificar quais são as deslocabilidade da estrutura que estão restringidas. Deste modo, esta informação será alocada através do índice do apoio no próprio objeto nó.

Como métodos, deveremos implementar os descritos no quadro 9.

4.2.6 Classe Nó

Um nó será definido por quatro características principais neste trabalho. Em primeiro lugar, teremos o seu *id*, que será utilizado para determinar o extremos das barras, um ponto C representando as suas coordenadas, a carga pontual Cp que nele atua e também as restrições de apoio R nele configuradas. Além disto, de modo a facilitar a montagem no modelo estrutural, também será inserida uma variável indicando se o nó está rotulado, situação para a qual se condicionará que $n - 1$ barras das n que nele se ligam

Quadro 9 – Tipo do membro em relação a sua rotulação

	Descrição	Entrada	Saída
getRestricoes	Retorna as restrições r_x, r_y, r_z a partir do índice i do apoio	**Inteiro** i	**Booleana** r_x, **Booleana** r_y, **Booleana** r_z
getApoioId	Retorna o índice i do apoio a partir de suas restrições r_x, r_y, r_z	**Booleana** r_x, **Booleana** r_y, **Booleana** r_z	**Inteiro** i

Fonte: autor

estejam com a devida extremidade rotulada, o que vai ao encontro de que foi definido por Kassimali (2012) na seção 2.2.3.

Como explicado anteriormente, para o caso de classes que estejam ligadas à objetos, serão utilizados índices para exprimir a relação entre os dois. Assim, no caso da carga pontual Cp, utilizaremos, na verdade a variável id atrelada a ela para representá-la.

No que tange os construtores utilizados nesta classe, preveremos dois: um que criará o nó a partir da definição de todas as suas variáveis constituintes e outro que o criará a partir de outro nó. Este último será especialmente importante nos casos onde será necessário dividir barras, por exemplo.

Quanto aos métodos, também serão previstos dois, para os quais temos a documentação explícita no quadro 10

Quadro 10 – Métodos da classe nó

	Descrição	Entrada	Saída
toString	Retorna uma representação inteligível em texto $desc$ do nó N	**Nó** N	**Texto** desc
getApNumero	Retorna o índice de apoio I apresentado pelo nó N	**Nó** N	**Inteiro** I

Fonte: autor

4.2.7 Classe Barra

A classe barra é um ótimo exemplo do que vem sendo falado sobre a referenciação através de índices. Por agrupar a maioria dos objetos que foram definidos até agora, se utilizará desta filosofia para indicar que $Material$, $Perfil$ e carga distribuída Q estão associados a ela, além de levar os índices dos nós que a extremam, possuindo também as variáveis $N1$ e $N2$. Ademais, também serão definidas outras características básicas delas, como o comprimento L, o seu â$ngulo$ e a sua vinculação nas extremidades rot.

Para as barras, será, como no caso dos nós, designado dois construtores. Para o primeiro, teremos a entrada de todas as variáveis, com exceção do seu ângulo e comprimento, os quais deverão ser calculados de acordo com as coordenadas dos dois nós que a delimitam. Para o segundo, será criada uma barra nova a partir de uma já existente, o que também é extremamente importante para casos onde há necessidade de se dividir barras.

No tocante aos métodos, nos ateremos a citar o *toString*, visto que os demais se referem ao funcionamento da interface programa-usuário.

4.2.8 Classe Estrutura

Por fim, temos a classe Estrutura. É nela que todas as operações referentes ao lançamento, alteração e cálculo da estrutura serão codificadas, fazendo com que deva conter todos os objetos já definidos, de modo que possa acessá-los de um modo simples. Logo, serão previstos vetores unidimensionais para cada um dos objetos, o que nos leva a obter. Adicionalmente, visto que o método dos deslocamentos será implementado nesta classe, também deveremos definir os objetos de matriz $mRigidez$, para a matriz de rigidez global, $mEngPerfeito$, para o vetor de forças de engastamento perfeito global, $mAcoesNodais$, para o vetor de ações nodais, e $mReacoesNodais$, para as reações nos apoios.

Nesta classe só será definido um construtor, já que no momento de criação do objeto estrutura, esta ainda estará vazia. Assim, o construtor não levará nenhuma variável de entrada além daquelas necessárias ao funcionamento do programa, como por exemplo, acesso ao banco de dados.

Por gerir todas as alterações na estrutura, esta classe contará com métodos mais operacionais em comparação com as demais, tal como adicionar uma barra, excluir um nó, etc. De todo modo, não se restringirá a eles, abrangendo todos representados no quadro 11.

Dos métodos apresentados, destacamos o método *calcular*, cujo fluxograma pode ser visto na figura 14. Sua elaboração foi feita com base nas diversas rotinas propostas por Kassimali (2012), realizando algumas adaptações no que diz respeito à criação de matrizes de rigidez e vetores de forças de engastamento perfeito.

4.3 Entrada de Dados

Uma vez definidos quais são as classes que constituirão o programa no âmbito teórico da análise, devemos definir também de que maneira será feita a entrada de dados por conta do usuário, para a qual adotaremos duas formas principais.

A inserção das barras será feita a partir da escolha de dois pontos na tela. Estes estarão atrelados a uma grelha de espaçamentos x e y configuráveis, que funcionará de

Quadro 11 – Métodos da classe Estrutura

	Descrição	Entrada	Saída
add...	Adiciona um novo elemento de carga, material ou perfil ao banco de dados da estrutura	**Variável**	**null**
addBarra	Adiciona uma nova barra com os extremos nos pontos A e B à estrutura identificando possíveis cruzamentos	**Ponto** A, **Ponto** B	**null**
exc...	Exclui do banco de dados da estrutura a carga, material ou perfil de índice *id*.	**Inteiro** id	**null**
excBarra	Exclui do banco de dados a barra de índice *id*	**Inteiro** id	**null**
excNo	Exclui do banco de dados o nó de índice *id*	**Inteiro** id	**null**
excNoSozinho	Identifica quais nós não têm barras associadas e os exclui.	**null**	**null**
populateAll	Acessa o banco de dados da estrutura e insere os dados nas respectivas variáveis.	**null**	**null**
calcular	Aplica o método dos deslocamentos na estrutura acessada.	**null**	**null**
getEsfBarra	Determina o vetor unidirecional de esforços $engPerfeito$ agindo na barra de índice i no seu sistema de coordenadas local	**Inteiro** i	**Número** engPerfeito
getDefBarra	Determina o vetor unidirecional de deslocamentos *desloc* da barra de índice i no seu sistema de coordenadas local	**Inteiro** i	**Número** desloc
checkNosRot	Verifica quais nós da estrutura são rotulados, de acordo com a condição estabelecida anteriormente	**null**	**null**

Fonte: autor

âncora para garantir que as suas coordenadas esteja, alinhadas. Com a criação destas barras, serão criados os nós inerentes, identificando também possíveis sobreposições de barras e nós já existentes. Uma vez adicionadas, será possível selecioná-las e então acessar uma área que, reunindo todas as informações intrínsecas às barras, possibilitará a alteração destas características. O mesmo poderá ser feito para os nós.

Os demais objetos, como apoios, materiais e perfis poderão ser configurados a partir de suas respectivas áreas, denotadas para Android de *fragments*. Estas áreas terão seu acesso garantido por um menu que compreenderá todas as funcionalidades do aplicativo e permitirão, além de inserir novos dados, excluir e editar os já existentes. A partir delas, também será possível atribuir características aos componentes da estrutura, ou seja, um determinado material a certas barras, ou ainda uma carga pontual a certos nós. Esta definição de características também poderá ser realizada para a estrutura como um todo, vis à vis da globalidade de certos aspectos dos elementos, como seu material de composição e a forma de sua seção.

4.4 Apresentação dos Resultados

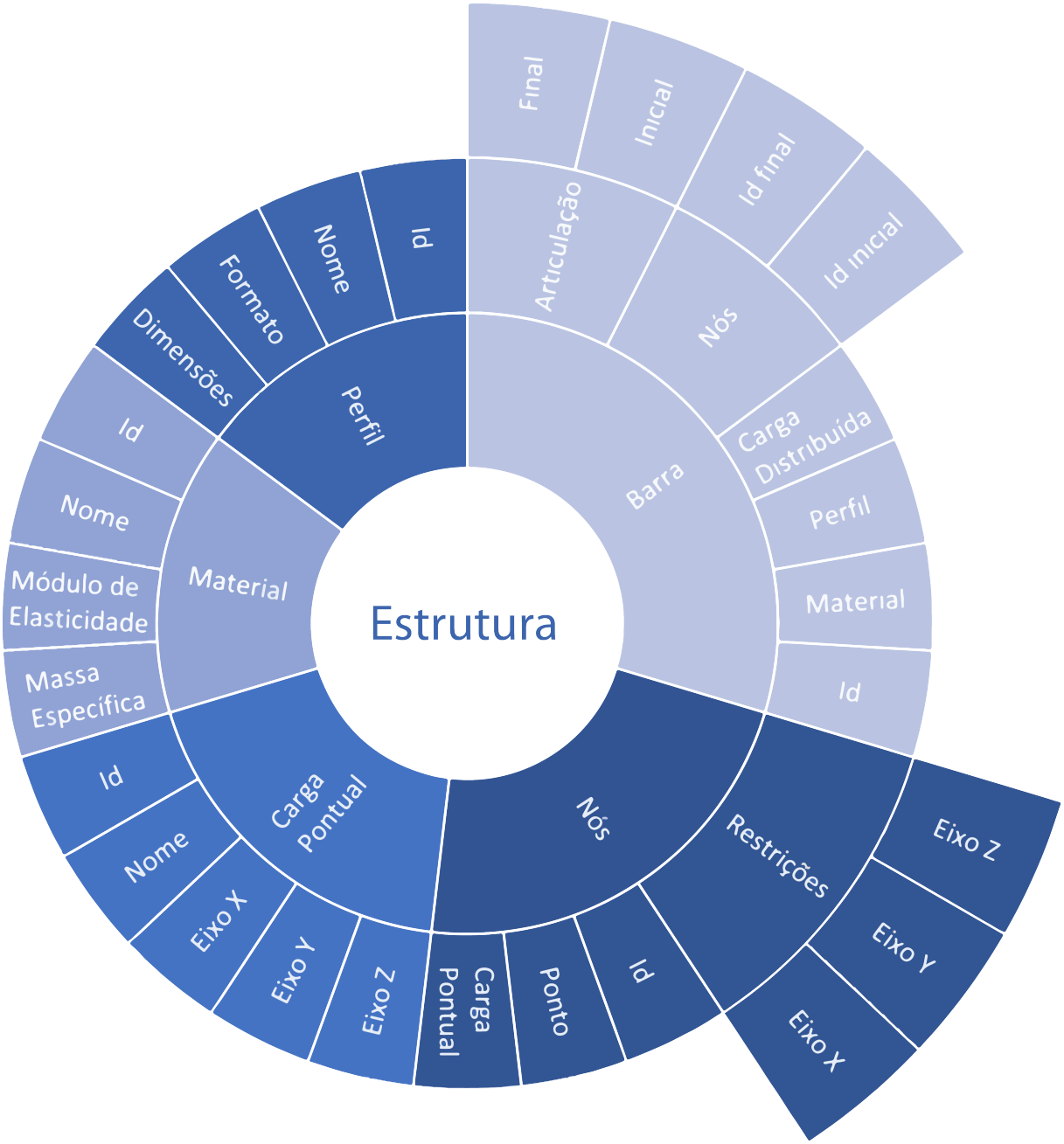
Por fim, devemos estabelecer como os resultados da análise serão apresentados ao usuário. Deve-se levar em consideração que estes são divididos em quatro etapas. Optou-se por manter o mesmo leiaute da entrada de dados. Todavia, há alguns pontos que foram adicionados de modo a facilitar o seu uso.

O primeiro e mais importante dentre eles é o fato de que os diagramas serão representados todos separadamente. Isto quer dizer que só será possível visualizar um diagrama por vez, sendo necessário para isso um comando que permita ao usuário trocar de uma representação para outra sem que o foco da tela se perca. Logo, optou-se por confeccionar um botão de rotina cíclica que alternasse entre as cinco representações: reações nodais, esforços normais, esforços cortantes, momentos fletores e estrutura deformada.

Outro ponto de grande valia para o bom funcionamento deste recurso é a escala variável. Esta ferramenta permite ao usuário a configuração de uma escala acordante com os diagramas representados em tela, o que contorna certos empecilhos como diagramas que extrapolam os limites da tela e valores que se sobrepõem, e ainda proporciona uma visão acentuada dos deslocamentos da estrutura, provando-se didática neste ponto.

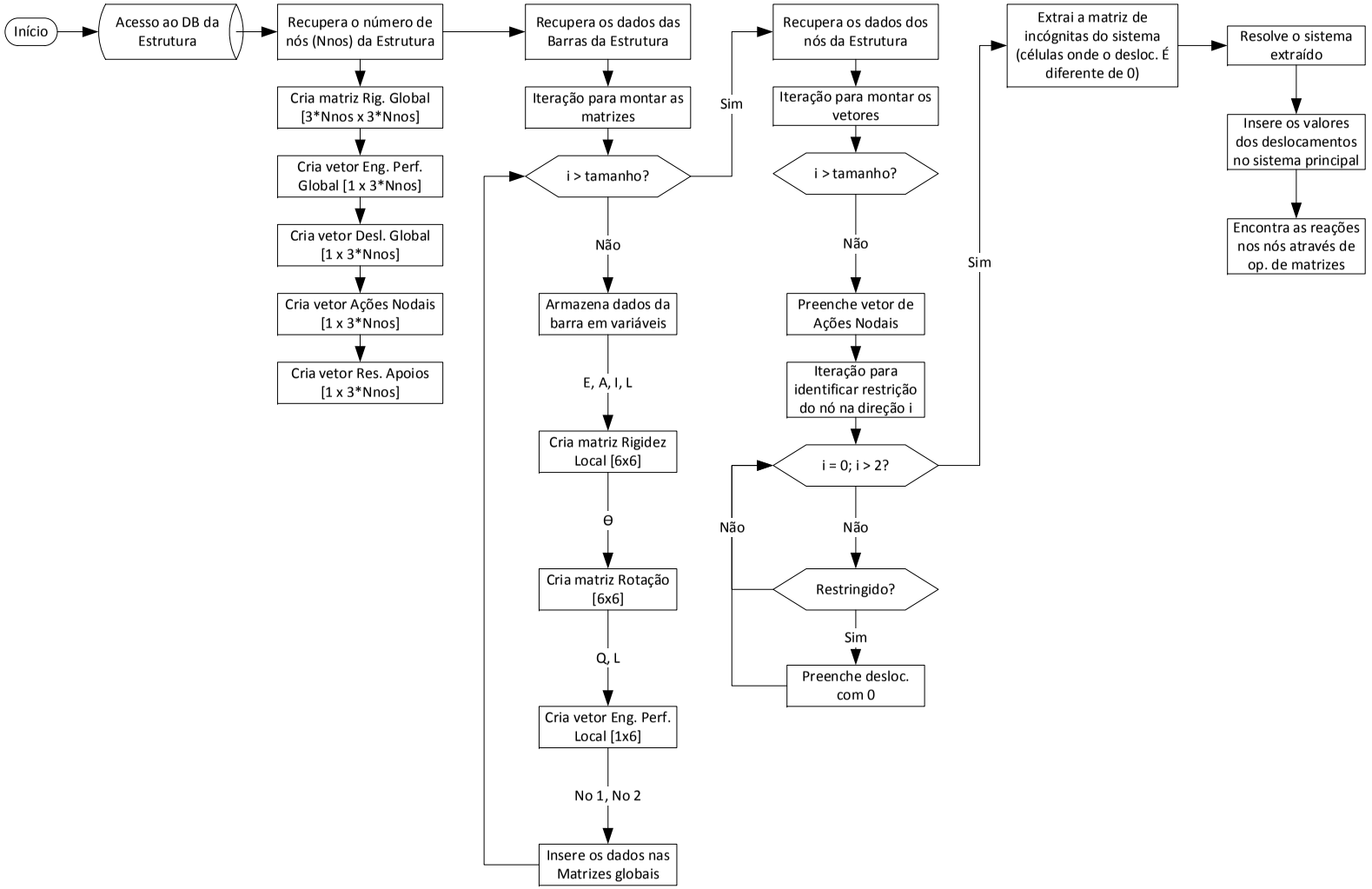
Figura 12 – Fluxograma eliminação gaussiana

Figura 13 – Organização hierárquica dos objetos constituintes da estrutura



Fonte: autor.

Figura 14 – Fluxograma de análise da estrutura



Fonte: autor.

5 Apresentação do Aplicativo

Como especificado no primeiro capítulo deste trabalho, a sua conclusão constará de um guia rápido de utilização para o aplicativo desenvolvido. Este guia reunirá todas as funções do programa, contando com o lançamento de um pórtico desde a configuração das características de seus componente até a visualização dos seus resultados.

5.1 Visão geral

A primeira tela, no momento de abertura, já apresenta algumas funcionalidades para a inserção dos elementos (Figura 15). Para aproveitar o máximo delas, é importante ter em mente que o aplicativo se vale de alguns gestos para facilitar a navegação. De um modo geral, para mover a tela, o usuário deverá tocar e arrastar. Já para ampliar ou reduzir a escala da estrutura, será utilizado o movimento de *pinch* (dois dedos na diagonal abrindo para ampliar e fechando para reduzir). Além disso, um toque duplo na tela fará com que a escala seja ajustada de modo a enquadrar toda a estrutura.

Esclarecidos estes pontos, podemos também perceber que três funções principais podem ser acessadas logo da tela de início.

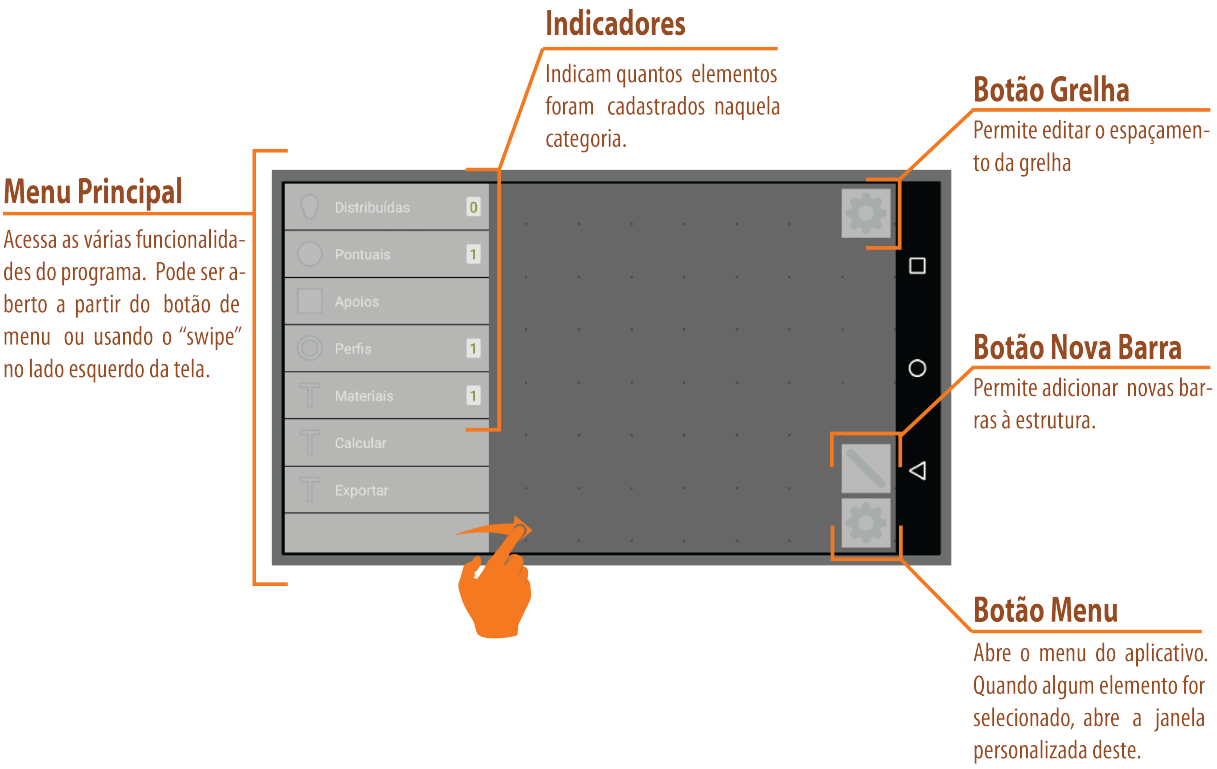
O botão barra permitirá a inserção de novas barras na estrutura, o que será visto posteriormente. Já o botão grelha acessa a janela de grelha, onde é possível editar os espaçamentos vertical e horizontal da mesma. Tendo em vista que nós extremos das barras somente poderão ser inseridos nos ponto de intersecção da grelha, esta ferramenta torna-se especialmente útil para casos onde é necessário um valor fracionário, por exemplo.

O botão menu permitirá ao usuário acessar o menu lateral do aplicativo, o qual contém, além dos elementos constituintes da estrutura, a opção de calculá-la. Este também pode ser aberto com um movimento de *swipe* para a direita e fechado com o mesmo movimento, porém à esquerda. Nele, também será possível verificar, através dos indicadores laterais, quantos itens de cada elemento já foram cadastrados pelo usuário. Podemos então averiguar cada uma destas janelas para identificar suas especificidades.

A janela aberta para o cadastro das cargas distribuídas pode ser visualizada na figura 16. É perceptível então que nesta janela pode-se gerenciar o cadastro, além de atribuir uma determinada carga distribuída para todas as barras da estrutura ou apenas para algumas específicas.

A janela aberta para o cadastro das cargas pontuais (Figura 17) é muito semelhante a das cargas distribuídas, divergindo apenas nos dados que são inseridos.

Figura 15 – Tela inicial do aplicativo



Fonte: autor.

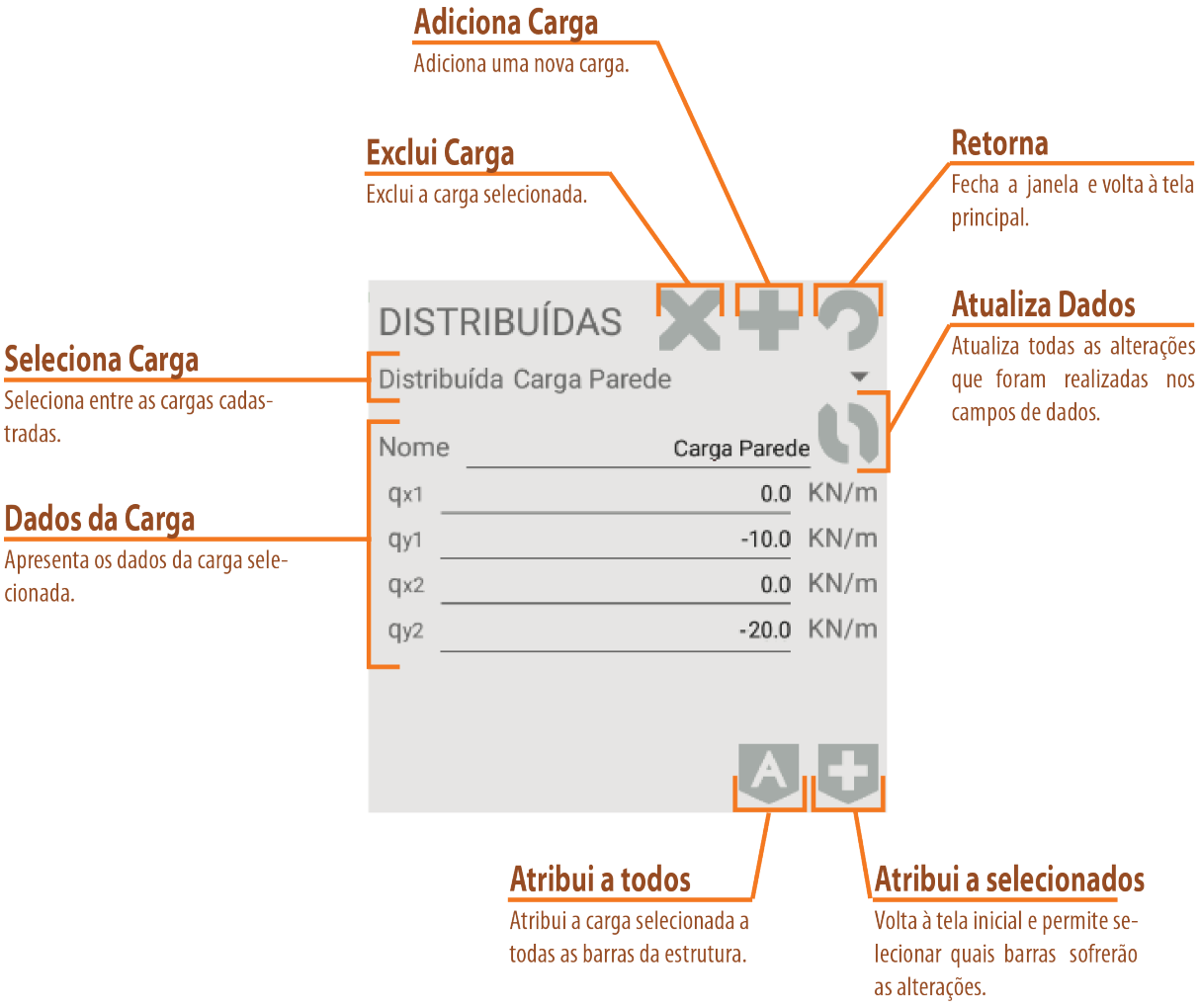
A janela de apoios, representada na figura 18 difere das anteriores, visto que nesta não há a necessidade de criar novos elementos. Sendo assim, esta tem o foco na definição das restrições dos nós da estrutura, contando também com uma representação gráfica dependendo das restrições selecionadas.

Apresentamos também a janela de materiais (Figura 19). Nela será possível cadastrar e gerir os materiais utilizados na confecção dos elementos da estrutura, para os quais será possível atribuir um módulo de elasticidade e uma massa específica

Por fim, a janela de perfis é exibida na figura 20. Como é mostrado, esta janela conta, além das ferramentas de controle já apresentadas, com a possibilidade de escolher o tipo de perfil que será utilizado (dentro dos definidos previamente) e também com uma representação gráfica em escala do perfil selecionado, que varia de acordo com as dimensões alteradas.

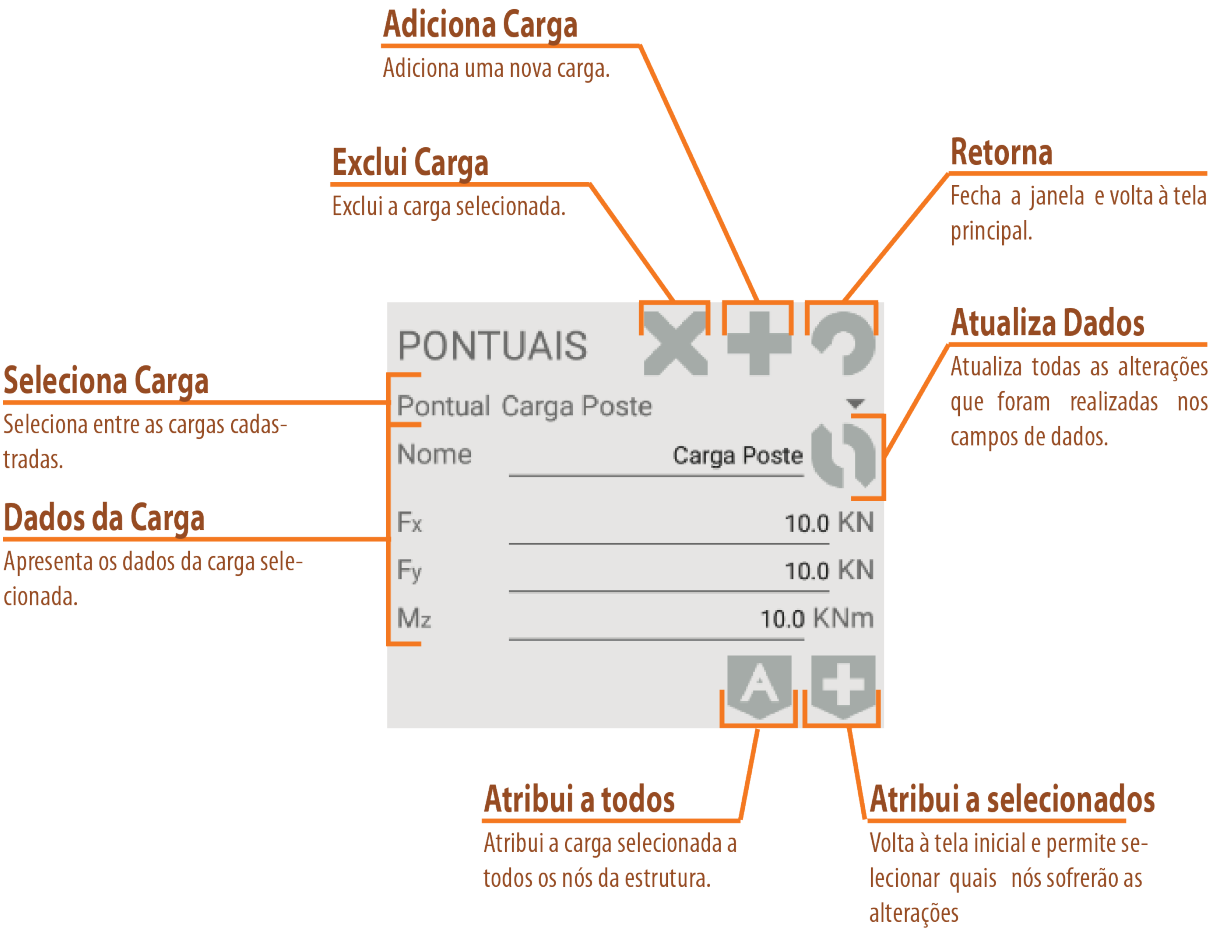
Também é interessante destacar que a codificação referente aos leiautes pode ser encontrada no apêndice E, no tocante à linguagem Java e no apêndice F, no tocante à estruturação *XML*.

Figura 16 – Cadastro de cargas distribuídas



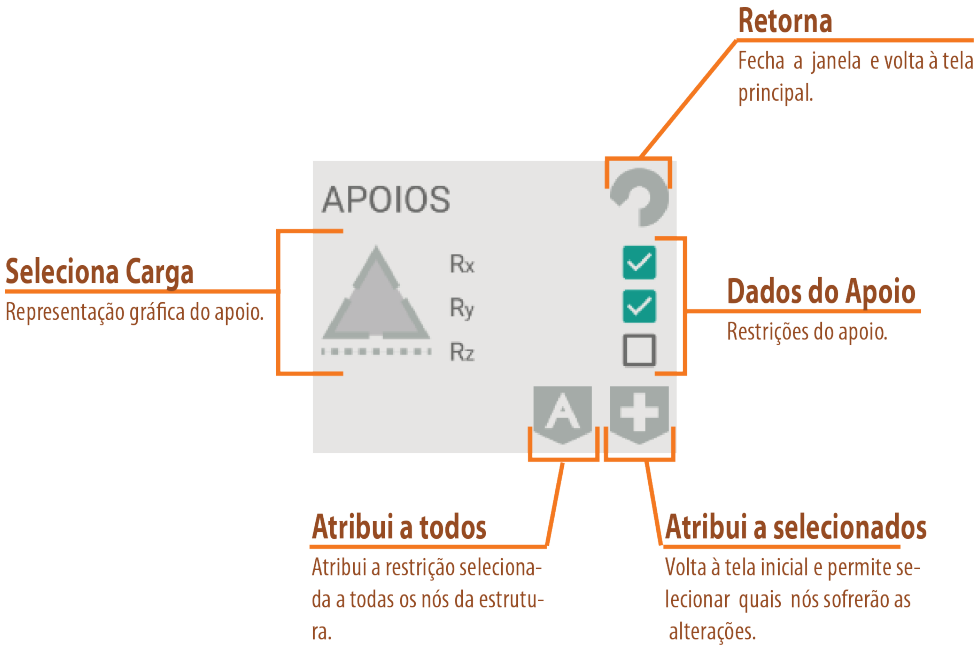
Fonte: autor.

Figura 17 – Cadastro de cargas pontuais



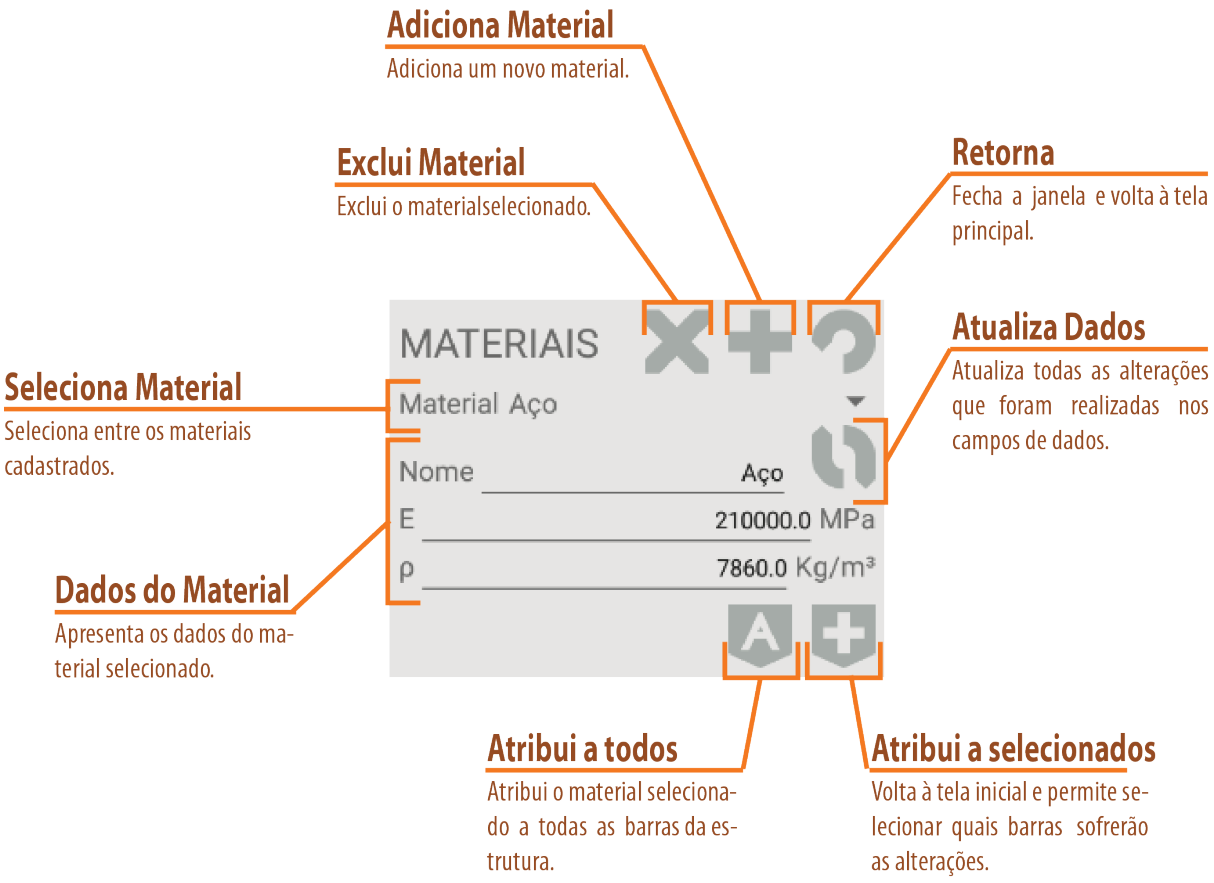
Fonte: autor.

Figura 18 – Apoios



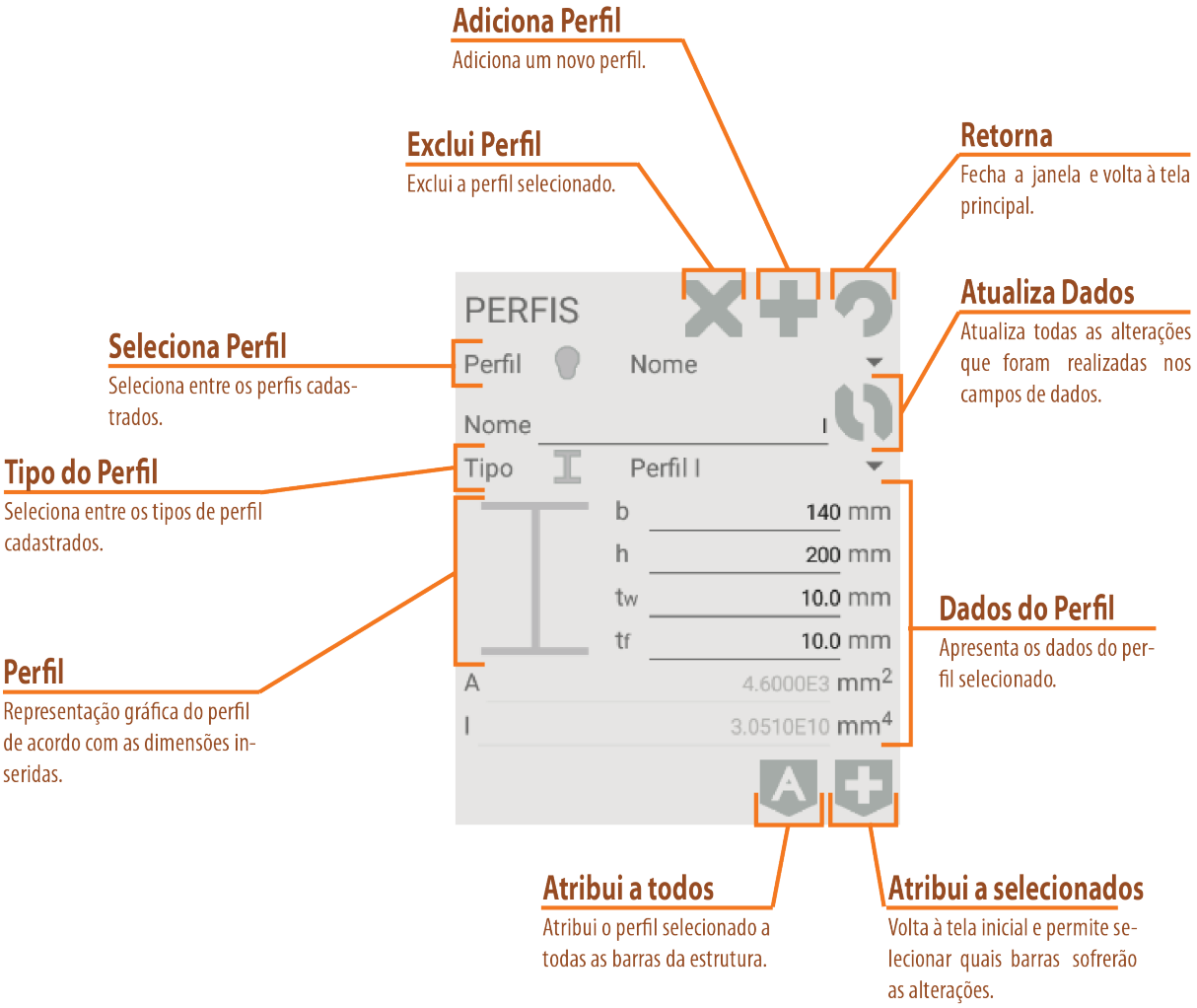
Fonte: autor.

Figura 19 – Cadastro de materiais



Fonte: autor.

Figura 20 – Cadastro de perfis



Fonte: autor.

5.2 Adicionar uma barra

A adição de barras está atrelada aos pontos da grelha configurados, isto é, os nós extremos das barras só poderão ser adicionados nestes pontos. Deste modo, é possível alterar o seu espaçamento de acordo com a necessidade da estrutura sendo construída.

Para acessar o modo "Adicionar Barra" no aplicativo, basta um clique no respectivo botão, representado anteriormente na figura 15. Feito isto, já será possível inseri-las, tendo em mente que isto será feito com um toque simples em cada uma das extremidades.

Outro aspecto do qual deve-se tomar nota, é o fato de que o aplicativo identifica pontos de cruzamento e insere um nó nele, não havendo necessidade de inserir as barras segmento por segmento. Todavia, este comportamento impede que sejam consideradas barras sobrepostas sem nenhuma ligação, como ocorre em alguns casos de contraventamento.

5.3 Excluir e alterar elementos já inseridos

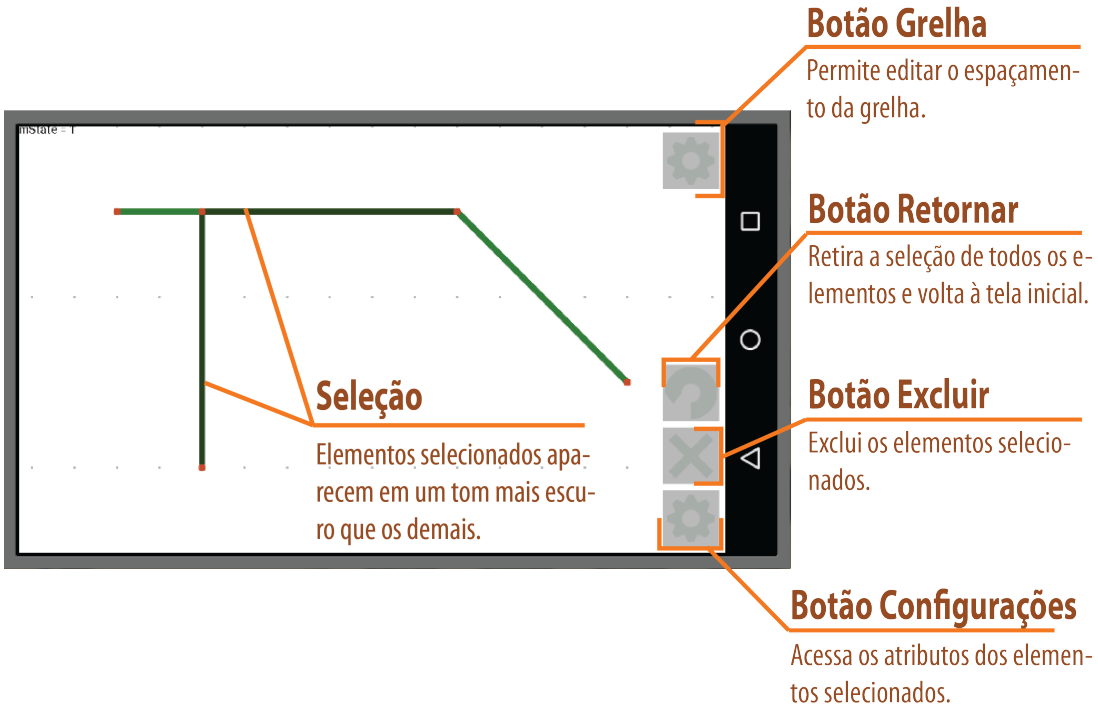
Em alguns casos, há a necessidade de alterar algumas características de um grupo de elementos já inseridos simultaneamente. Isto é possível com a seleção destes elementos através de um toque longo sobre um deles. Com este procedimento, o aplicativo entenderá que se deseja selecionar aquele elemento em questão e o marcará como tal, deixando-o num tom mais escuro, além de habilitar o modo de seleção. Uma vez nele, basta um toque simples para se selecionar os demais elementos.

Cabe ressaltar que esta seleção é exclusiva, isto é, tanto as barras quanto os nós podem ser selecionados, mas em momentos distintos. Tendo selecionado todos os elementos desejados, as alterações poderão ser feitas - neste momento, a tela estará organizada tal qual a representada na figura 21. Para isto, deve-se acessar os atributos dos elementos através do botão configurações, o qual abrirá uma janela referente ao tipo de elemento a ser alterado - nó ou barra

No caso de se alterarem nós, nesta janela serão apresentados as cargas pontuais atuantes, a rotulação do nó e também as suas restrições. Já para as barras, teremos as cargas distribuídas atuantes, seus perfis, seus materiais e também a vinculação em seus extremos.

Ainda em tempo, no modo seleção será possível excluir os elementos selecionados. No caso de exclusão de nós, todas as barras a ele ligadas serão apagadas, enquanto que no caso de exclusão de barras, todos os nós sem nenhuma barra a eles ligadas também serão excluídos.

Figura 21 – Selecionar elementos no aplicativo



Fonte: autor.

5.4 Resultados

Com a estrutura adicionada, os resultados poderão ser acessados a partir do item "Calcular"do menu principal. O programa tentará analisá-la de acordo com os métodos descritos neste trabalho, o que pode não ser possível por três motivos, para os quais serão emitidos avisos indicando isto:

a) Há elementos sem um perfil definido;

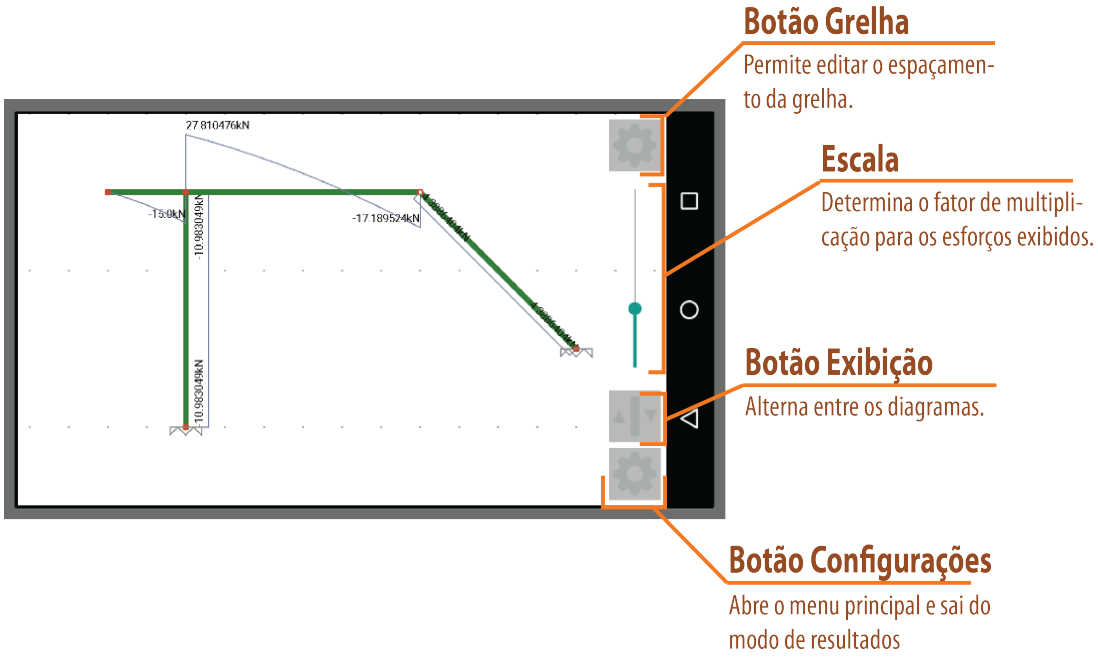
b) Há elementos sem um material definido;

c) A estrutura está hipostática.

Caso alguma destas três situações ocorra, o usuário deverá identificar aonde ela está ocorrendo e prever as alterações necessárias para que deixe de acontecer. Assim, uma vez que seja possível calcular a estrutura, o aplicativo mudará sua interface para a de exibição dos resultados, representada na figura 22.

Nesta tela, é perceptível que os resultados são apresentados separadamente, sendo que neste caso, mostramos os diagramas de momento fletor da estrutura em questão. Para alternar entre os resultados, o botão Exibição será utilizado, sendo que ele circulará entre as cinco possíveis representações: reações de apoio, esforços normais, esforços cortantes, momentos fletores e diagrama de deformações da estrutura. Além disso, também é possível

Figura 22 – Tela de resultados



Fonte: autor.

ajustar a escala destas representações, de modo a visualizar valores muito altos ou muito baixos mais facilmente.

Por fim, para voltar ao modo de inserção, caso se deseje fazer mais alguma alteração, basta acessar o menu principal da estrutura.

5.4.1 Validação dos Resultados

Como proposto na metodologia do trabalho, foram feitos testes para assegurar que os resultados apresentados pelo programa eram condizentes com a realidade, equivalendo àqueles da literatura. Esta validação poderá ser encontrada no apêndice C deste trabalho, sendo que será voltada para os resultados apresentados por ferramentas computacionais que operam no mesmo âmbito. Os pontos analisados para esta validação foram:

- a)Valor das reações nos apoios;
- b)Valor dos diagramas nos nós da estrutura;
- c)Formato dos diagramas;
- d)Formato da deformada;

5.5 Exemplo de cálculo: pórtico hiperestático

O primeiro passo a ser tomado para o cálculo da estrutura é inserir o seu corpo, isto é, suas características geométricas. Para tal deve-se clicar no botão adicionar barra (Figura 15). Adicionaremos quatro barras no pórtico, cujas coordenadas são listadas a seguir:

Barra 1 (1,0) e (1,2);

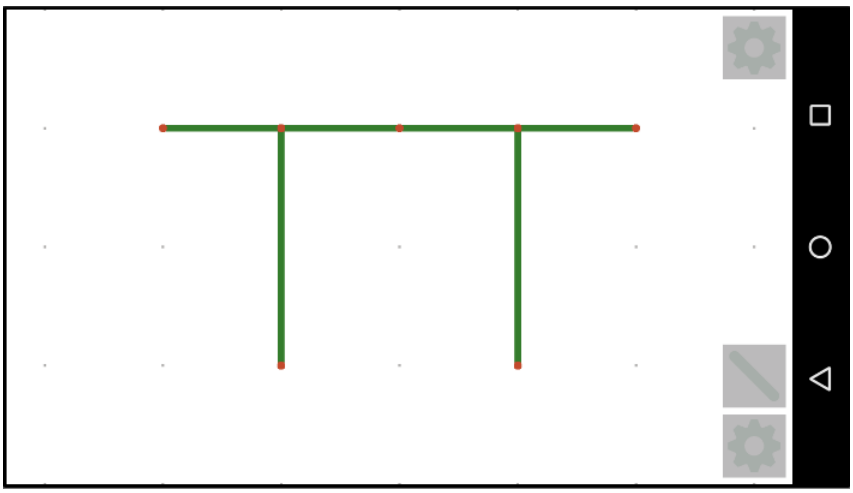
Barra 2 (3,0) e (3,2);

Barra 3 (0,2) e (2,2);

Barra 4 (2,2) e (4,2);

O lançamento deverá ficar igual ao da figura 23. Para o nó central, localizado nas coordenadas $(2, 2)$, vamos aplicar uma rótula, sendo necessário então selecioná-lo.

Figura 23 – Lançamento do pórtico exemplo

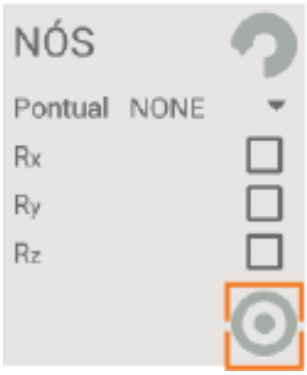


Fonte: autor.

Como já explicado, isto pode ser feito com um toque longo em cima do nó, após o qual a tela entrará no modo seleção. Clique então no botão "Configurações" (Figura 21) para acessar as configurações do nó. Agora, clique no botão rotular, como mostrado na figura 24. Já pode-se fechar esta janela.

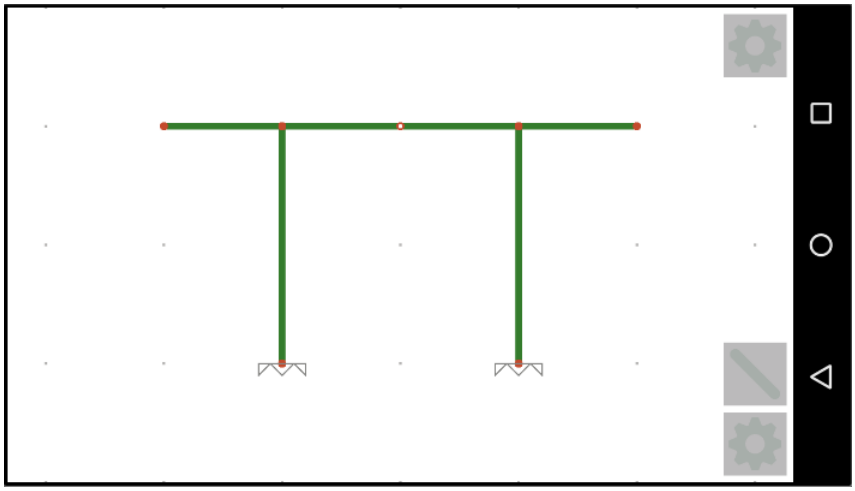
Com a rótula posicionada, devemos também prever os apoios da estrutura. Aqui, optaremos por engastar os dois nós inferiores do pórtico, para tanto selecionando ambos e clicando no botão "Configurações" novamente. Neste momento, marque as três opções de rotulação R_x, R_y e R_z. Perceba que neste momento, a figura demonstrativa do apoio tornou-se um engaste, bem como estes foram aplicados na estrutura. Assim, feche a janela. Neste momento, a estrutura deverá possuir as feições da figura 25.

Figura 24 – Rotulação do nó central



Fonte: autor.

Figura 25 – Lançamento da estrutura exemplo com restrições e articulações internas



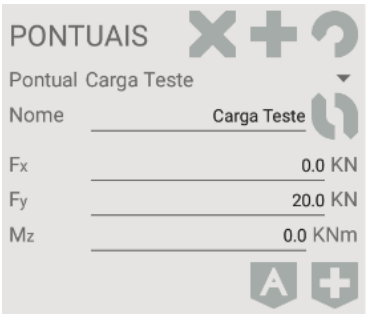
Fonte: autor.

Devemos agora aplicar o carregamento na estrutura, o que será feito em duas partes: admitiremos que toda a mesa superior do pórtico está submetida a uma carga linear de $10KN/m$ agindo de cima para baixo, sendo que os nós extremos (mais à esquerda e mais à direita) possuirão uma carga de $20KN$ agindo para cima. Deste modo, deveremos configurar cada uma destas.

Começando pela carga pontual, devemos acessar a sua janela através do botão de menu, clicando no item "Pontuais" (Figura 17). Clique no botão "Adicionar Carga" para criar uma carga nova e configure os valores apresentados na figura 26 para caracterizá-la. Então clique no botão "Atualizar dados" para confirmar as alterações. Por fim, clique no botão "Atribuir a selecionados", selecione os nós extremos e então clique no botão "Confirmar". Perceba que o programa já as adicionou aos pontos extremos e também já as representa atuando sobre eles.

Feito isto, podemos cadastrar a carga linear. Assim como a pontual, acesse o menu do programa, agora clicando no item "Distribuídas". Adicione uma nova carga com o botão "Adicionar Carga" (Figura 16). Configure-a tal qual os dados mostrados na figura

Figura 26 – Dados carga pontual



Fonte: autor.

27 e clique no botão “Atualizar dados” para efetivar as alterações. Então clique no botão “Atribuir a selecionados” e selecione todas as barras horizontais da estrutura, clicando então no botão confirmar.

Figura 27 – Dados carga distribuída

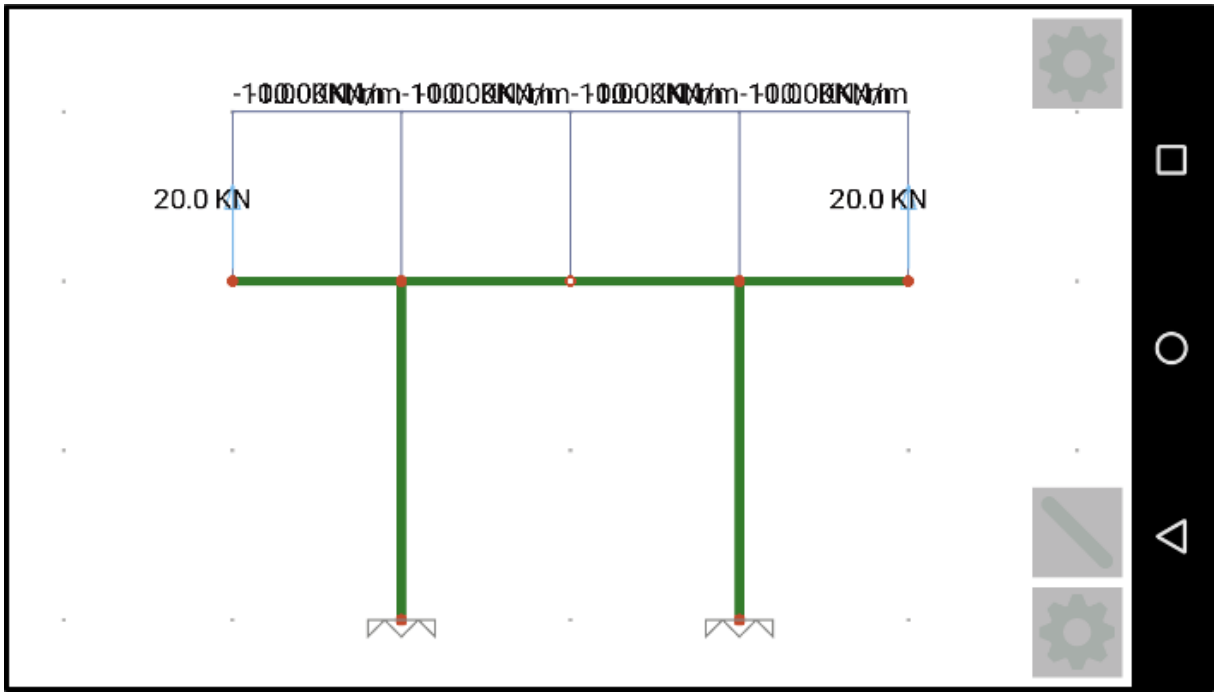
Fonte: autor.

Neste momento, tendo adicionado ambas as cargas na estrutura, já será possível visualizá-las no lançamento, tal como mostra a figura 28

Por fim devemos assegurar que os valores de inércia, área e módulo de elasticidade estão devidamente configurados. Assim, iremos cadastrar o material e a seção que serão utilizados. No que diz respeito, ao material, adotaremos uma liga genérica de aço, para a qual temos um módulo de elasticidade de $210GPa$. Do mesmo modo, a seção utilizada será o perfil I W250×44,8 (METALICAS, 2008), para a qual temos as seguintes dimensões: $h = 266mm$, $b = 148mm$, $t_w = 7,6mm$ e $t_f = 13,0mm$.

Deste modo, para acessar as características do material, clique no botão de menu na tela e então selecione o item Materiais. Já na janela, clique no botão "Adicionar Material"(Figura 19) e preencha-o com as características discriminadas na figura. Então, clique no botão “Atualizar Dados” para confirmar as alterações e, por fim no botão

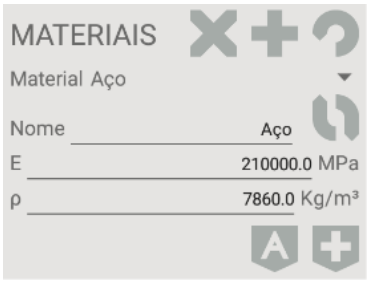
Figura 28 – Pórtico com cargas lançadas



Fonte: autor.

"Atribuir a todo" para atrelar todas as barras ao material configurado.

Figura 29 – Dados do material



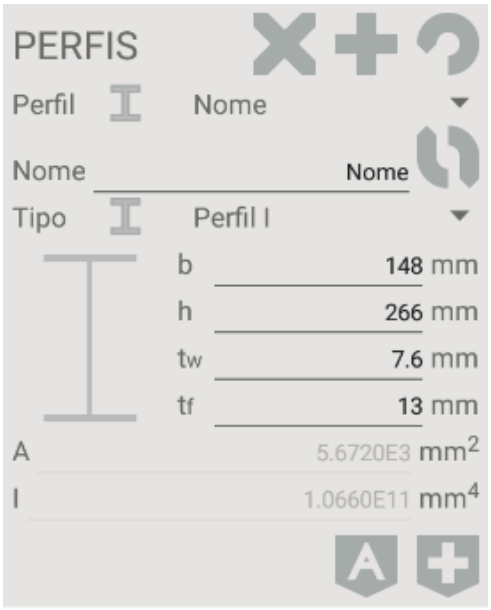
Fonte: autor.

O mesmo deverá ser feito para a seção das barras, sendo que nesse caso os dados inseridos serão os mostrados na figura 30.

Tendo configurado todos os aspectos da estrutura, já é possível visualizar seus resultados. Para isto, acesse o menu do aplicativo e clique em calcular. A estrutura será processada e assim que estiver calculada, serão exibidas as reações de apoio da mesma. Vale retomar que o tipo de resultado exibido poderá ser selecionado ciclicamente de acordo com o botão "Exibição" (Figura 22).

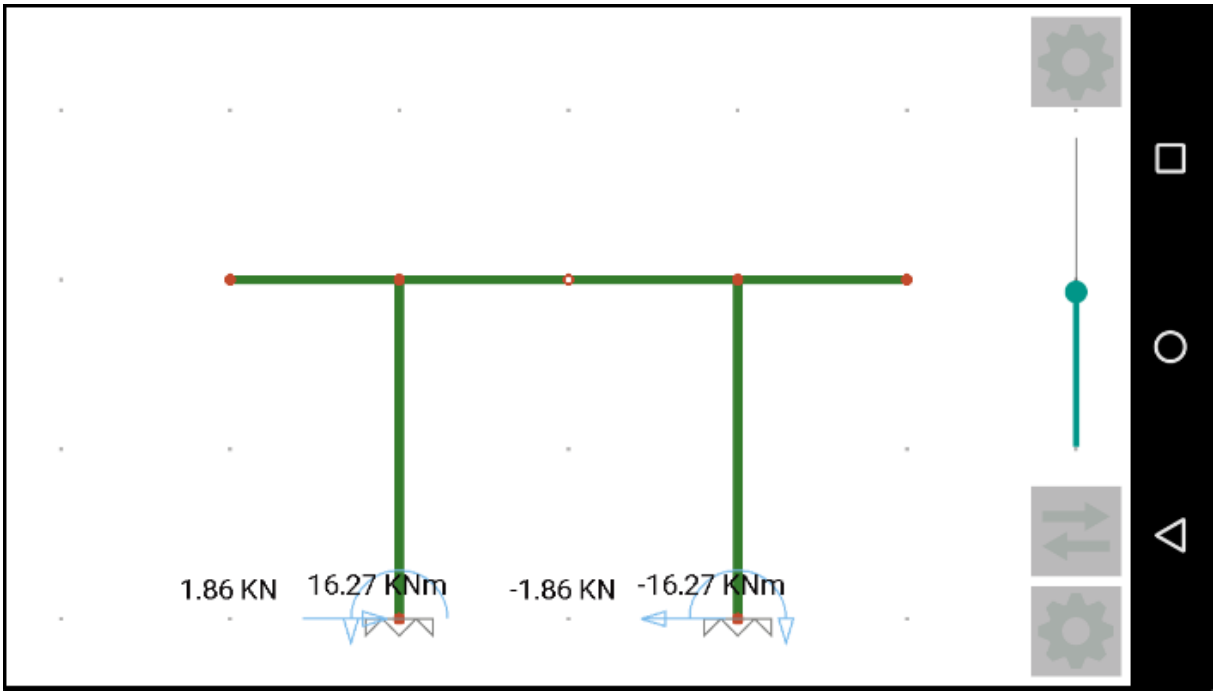
Os primeiros a serem exibidos serão as reações de apoio, para as quais temos os valores apresentados na figura

Figura 30 – Dados do material



Fonte: autor.

Figura 31 – Reações de apoio no pórtico

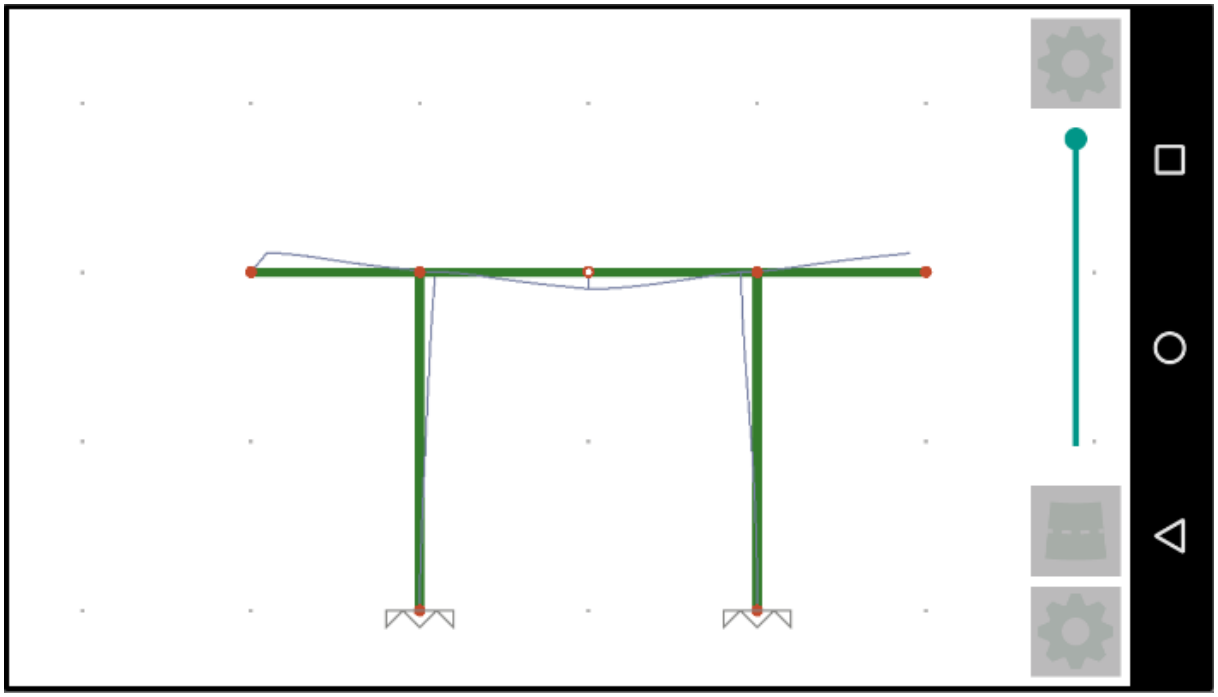


Fonte: autor.

Por conseguinte, a deformada da estrutura é apresentada (Figura 32). Também lembramos que a escala pode ser ajustada para uma melhor visualização dos deslocamentos.

O próximo diagrama representado é o de esforços axiais (Figura 33). Do mesmo modo que os deslocamentos, estes diagramas podem ser levados à escala mais conveniente para visualização. Note que as barras verticais do pórtico não apresentam esforços normais,

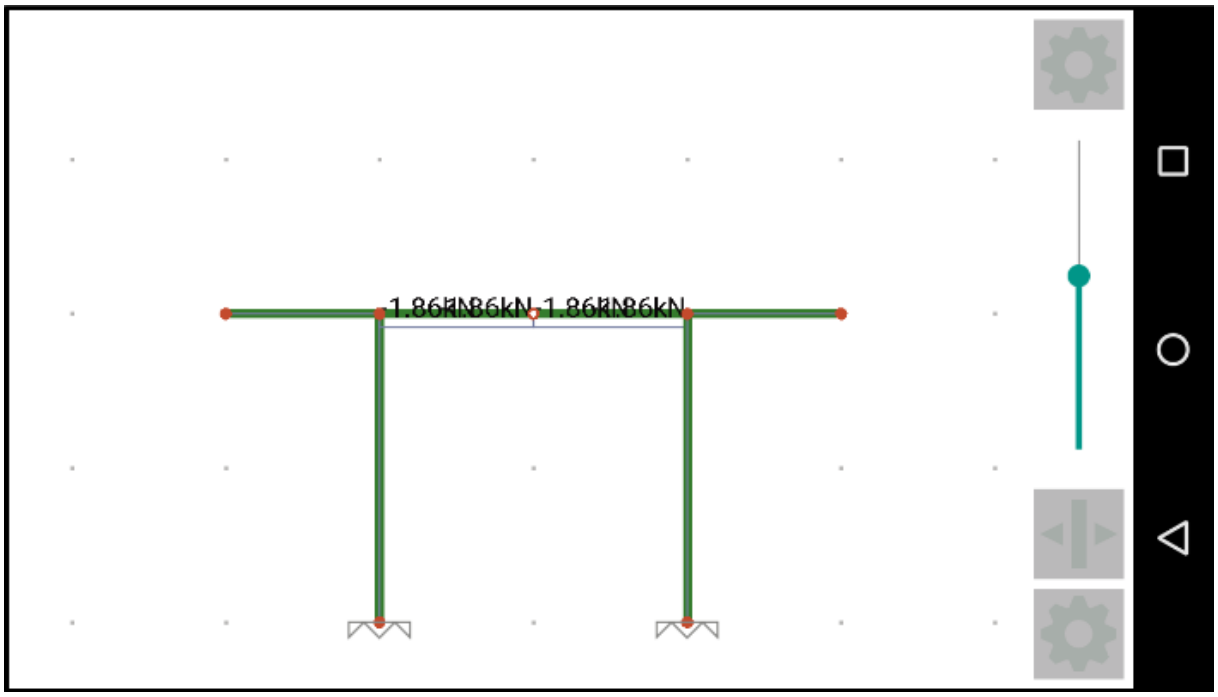
Figura 32 – Deslocamentos no pórtico



Fonte: autor.

visto que as forças aplicadas nas barras horizontais acabam se cancelando. Em compensação, as barras horizontais centrais apresentam um esforço de $1,86KN$, correspondente às reações horizontais nos apoios.

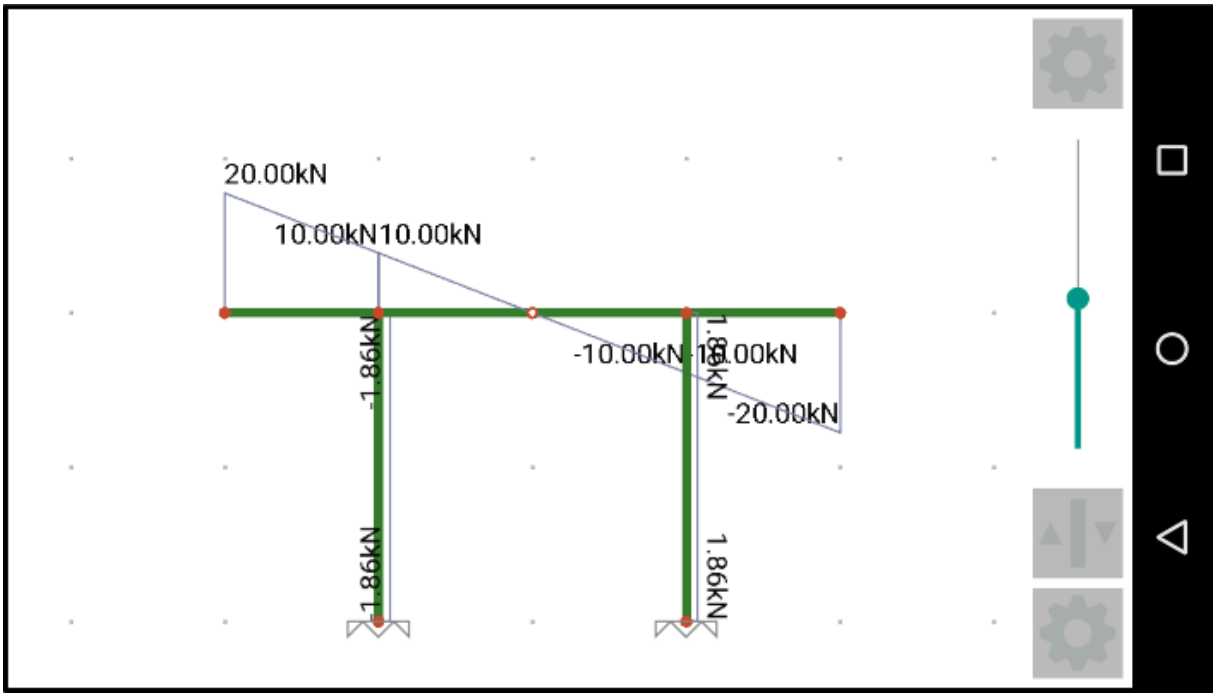
Figura 33 – Esforços axiais no pórtico



Fonte: autor.

Na sequência, poderemos analisar o diagrama de esforços cortantes (Figura 34). Nele, podemos perceber que as barras às quais foi atribuído o carregamento distribuído apresentam um diagrama linearmente variável, cujos extremos são compatíveis com as cargas pontuais de $20KN$ aplicadas. De encontro a isso, as barras verticais, sem carregamento ao longo do seu comprimento dispõem de um diagrama constante, de valor respectivo às cargas horizontais nos apoios.

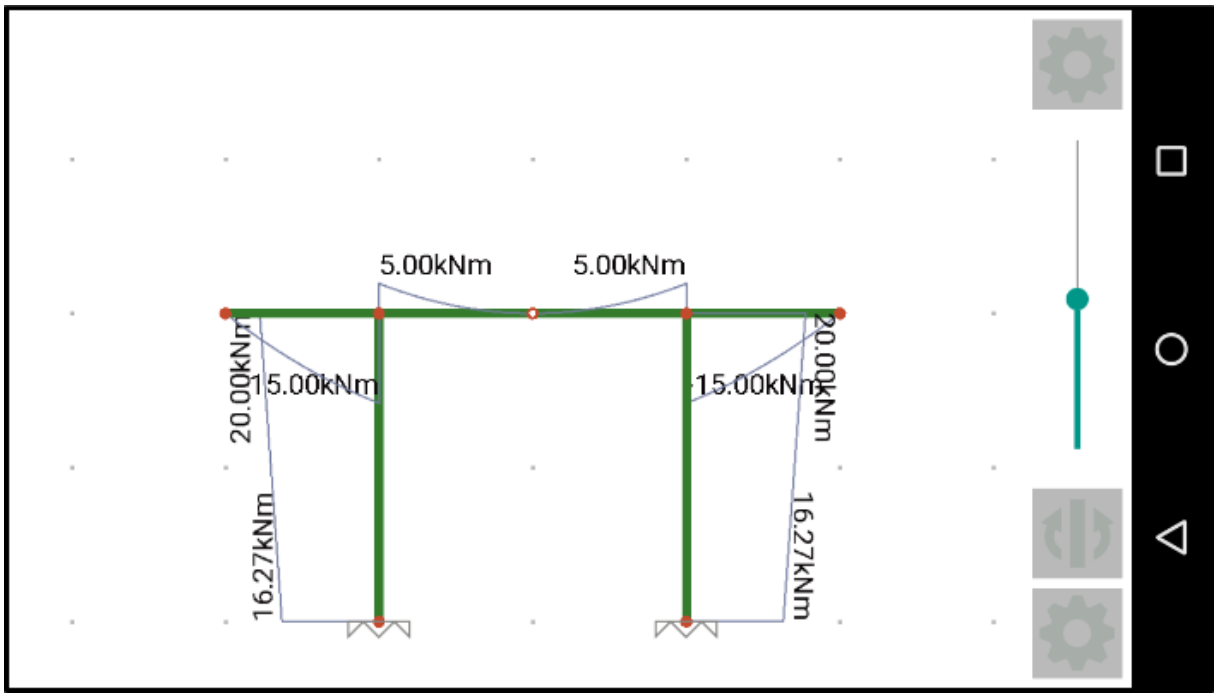
Figura 34 – Esforços cortantes no pórtico



Fonte: autor.

Finalmente, nos é apresentado o diagrama de momento fletores 35. Nele, é interessante notar que tanto as extremidades livres do pórtico quanto a rótula apresentam momentos nulos, comportamento que é esperado nessas regiões. Além disso, por possuírem uma carga linearmente distribuída constante, as barras horizontais apresentarão um diagrama de momento de formato parabólico, seguindo uma equação do segundo grau.

Figura 35 – Momentos fletores no pórtico



Fonte: autor.

6 Conclusão

Um dos pontos mais pronunciados do desenvolvimento de um produto é o caráter holístico que este assume. Com o decorrer do trabalho, percebeu-se que este processo engloba muito mais do que somente as ferramentas matemáticas e analíticas nas quais se baseia, ou mesmo a linguagem computacional que o compõe. Engloba também aspectos gráficos, de performance, além de uma série de conceitos inerentes à utilização de uma ferramenta portátil que buscam sempre uma interface mais intuitiva para o usuário.

Destarte, poderíamos nos demorar na análise da implementação do método dos deslocamentos e em quão apto ele está para o uso computacional. Contudo, visto que este é um lugar comum na engenharia de estruturas, é mais interessante que, salvoguardadas a sua importância e aptidão para a análise computacional de estruturas, seja avaliado o aplicativo como produto, sobretudo do ponto de vista de um usuário. Deste modo, ficam também indicadas possíveis melhorias e adaptações para versões futuras.

É importante que se compreenda que o desenvolvimento de um programa é um processo iterativo. Por conta disto, não é sensato admitir que a sua primeira versão será a mais eficiente, sendo este próprio aplicativo exemplo disto. Se colocarmos em paralelo os objetivos propostos no início do trabalho e o que foi alcançado no fechamento deste, facilmente se constatará que estes foram alcançados - todas as funcionalidades foram implementadas e o aplicativo desempenha o papel ao qual se propunha: ser uma ferramenta portátil para o aprendizado de análise estrutural.

Todavia, este tipo de comparação pode relevar alguns aspectos significativos do produto final, pois se limita ao escopo do projeto, não levando em consideração as possibilidades que nele não estão abrangidas. Estes aspectos possuem cunho variado, mas via de regra estarão atrelados à experiência do usuário com o aplicativo - cuja importância é vital no desenvolvimento de um programa de qualidade.

Entre eles, podemos destacar, por exemplo, o modelo de armazenamento dos dados da estrutura: até o momento, os dados da estrutura são diretamente armazenados em um banco de dados do próprio aplicativo, não sendo possível trabalhar com dois arquivos ao mesmo tempo - todas as alterações feitas nele são guardadas neste banco, impossibilitando a presença de dois arquivos. Além disso, algumas funcionalidades que abrangem em muito a versatilidade do aplicativo poderiam ter sido implementadas, como a consideração de cargas globais na estrutura, ou então a exibição das matrizes de rigidez da estrutura e da barra para o usuário, de modo que pudesse haver uma conferência de resultados ao longo do cálculo.

Alguns pontos da performance também devem ser melhorados, sobretudo no que

diz respeito a estruturas com um menor grau de restrição. Como visto no trabalho, quanto menor esse valor, maior será o sistema a ser resolvido, deixando logo o processamento mais lento. Deste modo, estes aspectos deverão ser considerados em uma próxima versão do programa, de modo que o aplicativo permita a análise de estruturas mais robustas.

No que diz respeito à interface do programa, também podemos destacar a inserção de barras que, num primeiro momento pode não ser óbvia. Ademais, os próprios aspectos gráficos do programa terão de ser melhorados, visto que a sua aparência ainda é áspera.

De todo modo, como dito anteriormente, este é um processo iterativo, sendo que versões futuras do programa poderão trazer estas correções. No âmbito positivo, foram implementados recursos interessantes que de fato diminuem o trabalho do usuário no que diz respeito ao lançamento, tal qual a identificação automática de cruzamento de barras ou a possibilidade de se atribuir características específicas a um conjunto de elementos ou então a todos eles.

Além disto, o aplicativo apresenta de um modo geral uma interface bem intuitiva, se utilizando dos diversos recursos que uma tela sensível ao toque oferece, destacando-se do conservadorismo comum a programas de engenharia, no que diz respeito a leiaute e interatividade.

6.1 Recomendações para trabalhos futuros

De acordo com o que foi realizado ao longo do trabalho, alguns pontos podem ser levantados como trabalhos futuros, podendo-se inclusive utilizar os códigos aqui desenvolvidos como base:

a) Desenvolver aplicativo para análise de grelhas, sendo possível também o dimensionamento de lajes através da analogia de grelha;

b) Implementar mais funcionalidades de cálculo a este aplicativo, como a possibilidade de inserir arcos, apoios deslocáveis, nós semirrígido, etc;

c) Implementar considerações de esforços de segunda ordem de acordo com os deslocamentos calculados;

d) Estender os códigos aqui desenvolvidos para outras plataformas, como *Windows Phone* e *iOS*

Referências

ATKINSON, K. E. *An introduction to numerical analysis*. 2. ed. New York: John Wiley and Sons, Inc, 1989. Citado 2 vezes nas páginas 47 e 56.

BEER, F. P. et al. *Vector Mechanics for Engineers*: Statics and dynamics. 9. ed. New York, Editora McGraw-Hill, 2010. Citado 2 vezes nas páginas 27 e 59.

BOLDRINI, J. L. et al. *Álgebra Linear*. 3. ed. Campinas, Editora HARBRA, 1980. Citado na página 35.

BRYANT, J. *Java 7 for Absolute Beginners*. 1. ed. New York: Springer Science+Bussines Media, 2012. Citado na página 49.

GHALI, A.; BROWN, T. G.; NEVILLE, A. M. *Structural Analysis*: A unified classical and matrix approach. 6. ed. New York, Spon Press, 2009. Citado 5 vezes nas páginas 27, 28, 35, 36 e 52.

GLOBAL WEB INDEX. *Gwd Device Summay*: Globalwebindex's quarterly report on the latest trends for smartphones, tablets, smart tvs and wearables. c2005. Disponível em: <http://www.globalwebindex.net/blog/80-of-internet-users-own-a-smartphonep>. Acesso em: 19 jun. 2015. Citado na página 21.

ISO 5807:1985: Information processing - documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts. [S.l.]. Citado na página 53.

JACKSON, W. *Android Apps for Absolute Beginners*. 1. ed. New York: Springer Science+Bussines Media, 2011. Citado 2 vezes nas páginas 48 e 49.

KASSIMALI, A. Matrix analysis of structures. Stamford: Cengage Learning, 2012. Citado 11 vezes nas páginas 26, 32, 33, 36, 39, 41, 42, 46, 52, 61 e 62.

MARTHA, L. F. *Análise de Estruturas*. Rio de Janeiro, Editora Campus/Elsevier, 2010. v. 1, 524 p. Citado 10 vezes nas páginas 26, 27, 28, 29, 30, 31, 32, 33, 42 e 52.

MCGUIRE, W.; GALLAGHER, R. H.; ZIEMIAN, R. D. *Matrix Structural Analysis*. 2. ed. New York: John Wiley and Sons, Inc, 2000. Citado 2 vezes nas páginas 46 e 52.

METALICAS, S. E. *Tabela de perfis Laminados I e H*. 2008. Disponível em: <http://www.skylightestruturas.com.br/perfis_ih.asp>. Acesso em: 11 nov. 2015. Citado na página 80.

Project Tomorrow. *Creating our future*: Students speak up about their vision for 21st learning. 2009. 2009. Disponível em: <http://www.tomorrow.org/speakup/pdfs/su09NationalFindingsStudentsParents;.pdf>. Acesso em: 03 jul. 2015. Citado 2 vezes nas páginas 21 e 22.

STEINBRUCH, A.; WINTERLE, P. *Geometria Analítica*. 2. ed. São Paulo: Makron, 1987. Citado na página 54.

STEINBRUCH, A.; WINTERLE, P. *Álgebra Linear*. 2. ed. São Paulo: Makron, 1987. Citado na página 56.

SUSSEKIND, J. C. *Curso de análise estrutural*. Rio de Janeiro, Editora Globo, 1981. v. 1, 366 p. Citado 6 vezes nas páginas 25, 26, 31, 46, 52 e 99.

SUSSEKIND, J. C. *Curso de análise estrutural*. 7. ed. Rio de Janeiro, Editora Globo, 1987. v. 3, 294 p. Citado 5 vezes nas páginas 27, 30, 34, 95 e 97.

Apêndices

APÊNDICE A – Cálculo das Forças de Engastamento Perfeito

A.1 Caracterização da carga distribuída

Visto que as cargas distribuídas consideradas nestes trabalho são de distribuição linear, devemos definir uma equação que as modelem. Do mesmo modo, visto que as cargas atuam tanto ao longo das barras quanto transversalmente a elas, o carregamento pode ser representado como um vetor cujas coordenadas são definidas em função de um ponto x ao longo da barra (A.1). Cabe ainda ressaltar que as componentes do vetor carregamento estão atreladas ao sistema de coordenadas locais, x sendo, então, o eixo da barra e y o eixo transversal a esta.

$$\vec{q}(x) = (q_x(x), q_y(x)) \tag{A.1}$$

Se tratando de uma distribuição linear, podemos ainda expressar o vetor $q(\vec{x})$ na forma da equação A.2

$$\vec{q}(x) = \left(q_{x0}\left(1 - \frac{x}{L}\right) + q_{x1}\left(\frac{x}{L}\right), q_{y0}\left(1 - \frac{x}{L}\right) + q_{y1}\left(\frac{x}{L}\right)\right) \tag{A.2}$$

, onde:

q_{x0}: Valor da carga distribuída na direção x no nó inicial da barra

q_{y0}: Valor da carga distribuída na direção y no nó inicial da barra

q_{x1}: Valor da carga distribuída na direção x no nó final da barra

q_{y1}: Valor da carga distribuída na direção y no nó final da barra

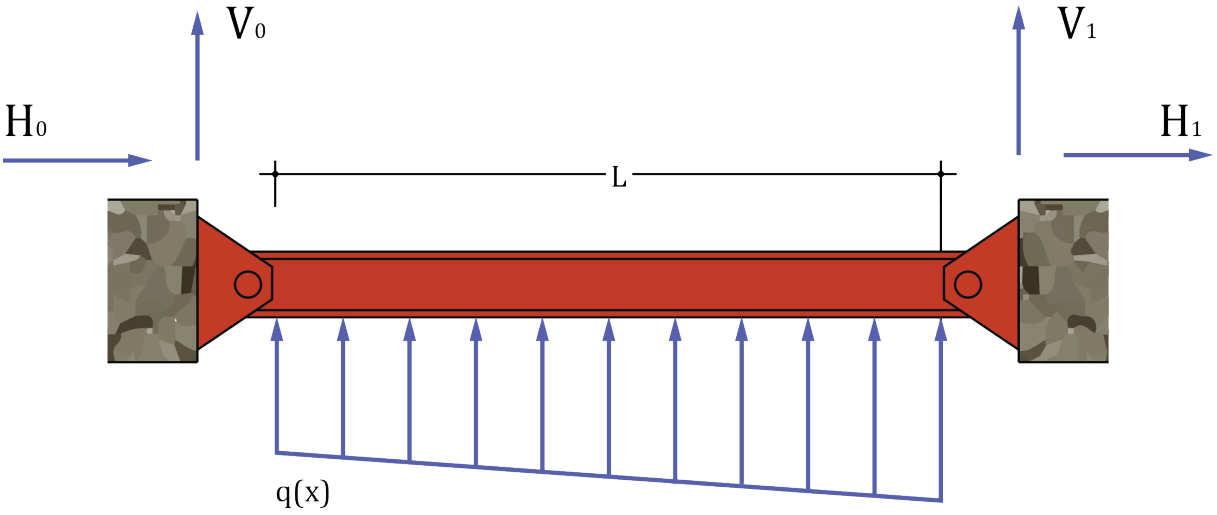
x: Ponto da barra, a contar do nó inicial, onde se mede o valor da carga

L: Comprimento total da barra

A.2 Membro do tipo 3 (MT = 3)

Como definido no trabalho, um membro do tipo quatro apresenta ambas as suas extremidades livres, implicando na inexistência de momentos nas mesmas. Sendo assim, podemos ilustrar esta condição com a figura 36:

Figura 36 – Barra MT = 3 com carregamento linearmente variável



Fonte: autor.

Nota-se também que o sentido das forças na figura 36 foi escolhido de modo a coincidir com o positivo da convenção definida na seção 2.2.5, não representando necessariamente o real sentido das mesmas.

Para definir os valores das reações de apoio, podemos nos utilizar das equações de equilíbrio estático. Aplicando-as, obtemos:

$$\sum V = 0 \rightarrow V_0 + V_1 + \int_0^L q_y(x)dx = 0 \rightarrow V_0 + V_1 = -\int_0^L q_{y0}\left(1 - \frac{x}{L}\right) + q_{y1}\left(\frac{x}{L}\right)dx$$

$$V_0 + V_1 = -\left[q_{y0}\left(x - \frac{x^2}{2L}\right) + q_{y1}\left(\frac{x^2}{2L}\right)\right]_0^L$$

$$V_0 + V_1 = -\frac{1}{2}\Big[q_{y0} + q_{y1}\Big]L$$

$$\sum M_1 = 0 \rightarrow -V_0 L - \iint_0^L q_y(x)dx = 0 \rightarrow V_0 L = -\frac{1}{L}\iint_0^L q_{y0}\left(1 - \frac{x}{L}\right) + q_{y1}\left(\frac{x}{L}\right)dx$$

$$V_0 = -\frac{1}{L}\left[q_{y0}\left(\frac{x^2}{2} - \frac{x^3}{6L}\right) + q_{y0}\left(\frac{x^3}{6L}\right)\right]_0^L$$

$$V_0 = -\frac{1}{6}\Big[2q_{y0} + q_{y1}\Big]L$$

$$V_1 = -\frac{1}{6}\Big[q_{y0} + 2q_{y1}\Big]L$$

$$\sum N = 0 \rightarrow H_0 + H_1 + \int_0^L q_x(x)dx = 0 \rightarrow H_0 + H_1 = -\int_0^L q_{x0}\left(1 - \frac{x}{L}\right) + q_{x1}\left(\frac{x}{L}\right)dx$$

Das condições de contorno, temos que $\delta L = 0$. Aliado a isso, da elasticidade:

$$\delta L = \frac{1}{EA}\int_0^L N(x)dx = \frac{1}{EA}\int_0^L -H_0 - \left(\int_0^x q_{x0}\left(1 - \frac{x}{L}\right) + q_{x1}\left(\frac{x}{L}\right)dx\right)dx = 0$$

$$\left[-H_0 x - q_{x0}\left(\frac{x^2}{2} - \frac{x^3}{6L}\right) - q_{x1}\left(\frac{x^3}{6L}\right)\right]_0^L = 0$$

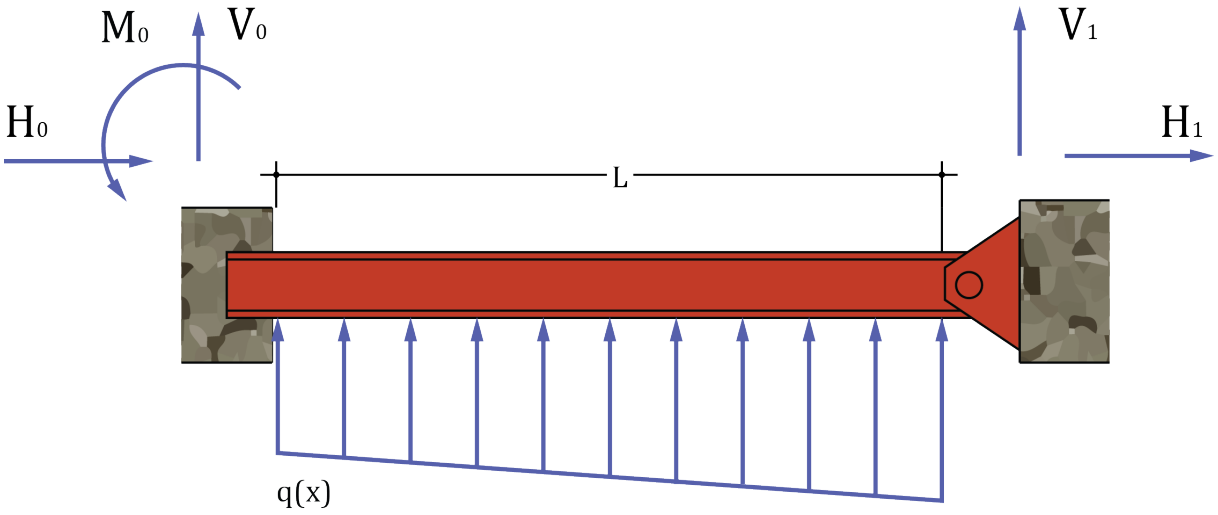
$$H_0 = -\frac{1}{6}\Big[2q_{x0} + q_{x1}\Big]L$$

$$H_1 = -\frac{1}{6}\Big[q_{x0} + 2q_{x1}\Big]L$$

A.3 Membro do tipo 2 (MT = 2)

Para elementos do tipo 2, temos que a extremidade final da barra será rotulada, como mostrado na figura 37.

Figura 37 – Barra MT = 2 com carregamento linearmente variável



Fonte: autor.

Para o momento M_0 na extremidade inicial, podemos consultar Sussekind (1987), que admite, para uma barra rotulada em uma das extremidades e submetida a um carregamento de carga variável, aqui encarado como a soma de dois carregamentos triangulares, um valor de:

$$M_0 = -\frac{1}{15}q_{y0}L^2 - \frac{7}{120}q_{y1}L^2 = -\frac{1}{120}\left(8q_{y0} + 7q_{y1}\right)L^2$$

Podemos nos utilizar do que foi determinado na seção A.2 deste apêndice. Como temos condições de contorno similares, podemos afirmar que:

$$\begin{aligned} H_0 &= -\frac{1}{6}\Big[2q_{x0} + q_{x1}\Big]L \\ H_1 &= -\frac{1}{6}\Big[q_{x0} + 2q_{x1}\Big]L \\ V_0 + V_1 &= -\frac{1}{2}\Big[q_{y0} + q_{y1}\Big]L \end{aligned}$$

Ainda, da estática, podemos fazer a soma dos momentos fletores no nó final da estrutura, que deverá ser igual a 0:

$$\sum M_1 = 0 \rightarrow M_0 - V_0L - \iint_0^L q_y(x)dx \rightarrow V_0L = -M_0 + \iint_0^L q_{y0}\left(1 - \frac{x}{L}\right) + q_{y1}\left(\frac{x}{L}\right)dx$$

$$V_0L = M_0 + \left[q_{y0}\left(\frac{x^2}{2} - \frac{x^3}{6L}\right) + q_{y1}\left(\frac{x^3}{6L}\right)\right]_0^L$$

$$V_0L = -\frac{1}{120}\left(8q_{y0} + 7q_{y1}\right)L^2 - \frac{1}{6}\left(2q_{y0} + q_{y1}\right)L^2$$

$$V_0L = -\frac{1}{120}\left(48q_{y0} + 27q_{y1}\right)L^2$$

$$V_0 = -\frac{1}{120}\left(48q_{y0} + 27q_{y1}\right)L$$

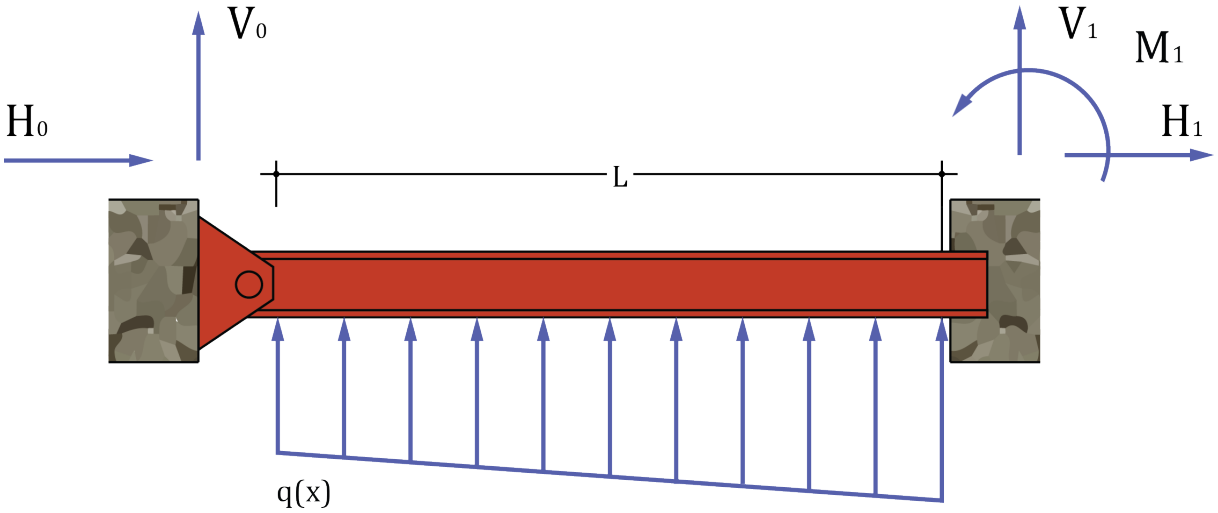
$$V_1 = -\frac{1}{120}\left(12q_{y0} + 33q_{y1}\right)L$$

A.4 Membros do tipo 1 (MT = 1)

Membros do tipo 1 são bastante similares aos do tipo 2, com a única ressalva de que, ao invés de apresentarem a sua extremidade final rotulada, apresentam a sua extremidade inicial (Figura 38).

Com isto em mente, podemos utilizar o que foi deduzido na seção A.3 para definir as forças de engastamento perfeito desta classe de membros, sempre levando em consideração a convenção de sinais:

Figura 38 – Barra MT = 1 com carregamento linearmente variável



Fonte: autor.

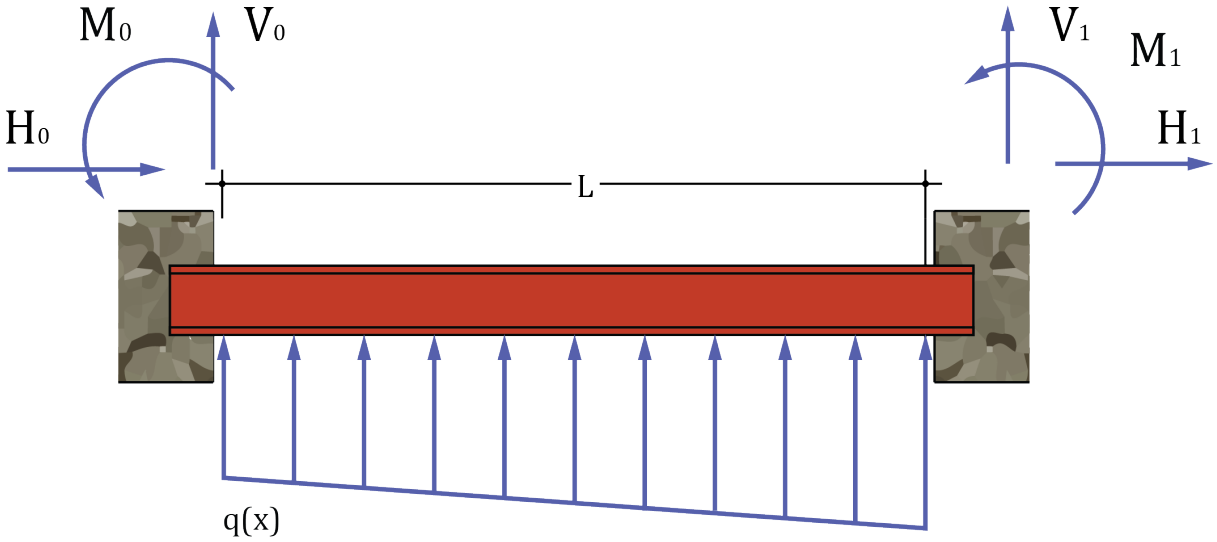
$$
\begin{aligned}
H_0 &= -\frac{1}{6}\Big[2q_{x0} + q_{x1}\Big]L \\
H_1 &= -\frac{1}{6}\Big[q_{x0} + 2q_{x1}\Big]L \\
V_0 &= -\frac{1}{120}\left(33q_{y0} + 12q_{y1}\right)L \\
V_1 &= -\frac{1}{120}\left(27q_{y0} + 48q_{y1}\right)L \\
M_1 &= \frac{1}{120}\left(7q_{y0} + 8q_{y1}\right)L^2
\end{aligned}
$$

A.5 Membro do tipo 0 (MT = 0)

Estes membros não possuirão articulações em suas extremidades, se tratando, portanto de barras biengastadas, como mostra a figura 39.

Temos, dos demais casos a expressão para a soma das forças verticais na barra e a expressão para as forças horizontais. Já para determinação dos momentos causados pelos apoios de terceiro grau, recorremos novamente a Sussekind (1987). Destes dois pontos, obtemos que:

Figura 39 – Barra MT = 0 com carregamento linearmente variável



Fonte: autor.

$$
\begin{aligned}
H_0 &= -\frac{1}{6}\Big[2q_{x0} + q_{x1}\Big]L \\
H_1 &= -\frac{1}{6}\Big[q_{x0} + 2q_{x1}\Big]L \\
M_0 &= -\frac{1}{120}\left(6q_{y0} + 4q_{y1}\right)L^2 \\
M_1 &= \frac{1}{120}\left(4q_{y0} + 6q_{y1}\right)L^2 \\
V_0 + V_1 &= -\frac{1}{2}\Big[q_{y0} + q_{y1}\Big]L
\end{aligned}
$$

Fazemos a soma de momentos no nó final da barra, de onde tiramos:

$$
\begin{aligned}
\sum M_1 = 0 \rightarrow M_0 + M_1 - V_0 L - \iint_0^L q_y(x)dx = 0 \rightarrow V_0 L &= M_0 + M_1 - \iint_0^L q_y dx \\
V_0 L &= M_0 + M_1 - \left[q_{y0}\left(\frac{x^2}{2} - \frac{x^3}{3L}\right) + q_{y1}\left(\frac{x}{L}\right)\right]_0^L \\
V_0 L &= -\frac{1}{120}\left(6q_{y0} + 4q_{y1}\right) + \frac{1}{120}\left(4q_{y0} + 6q_{y1}\right) - \frac{1}{6}\left(2q_{y0} + q_{y1}\right)L^2 zz \\
V_0 &= -\frac{1}{20}\left(7q_{y0} + 3q_{y1}\right)L \\
V_1 &= -\frac{1}{20}\left(3q_{y0} + 7q_{y1}\right)L
\end{aligned}
$$

APÊNDICE B – Determinação das equações dos diagramas da barra

Para a determinação dos diagramas, partiremos de três equações (Equação B.1) citadas por Sussekind (1981). Para isto, deveremos resolvê-las, aplicando também as condições de contorno das barras. Cabe ressaltar que as considerações previstas no apêndice A são todas válidas, contudo não haverá necessidade de distinguir entre os tipos de membro MT, visto que virtualmente as características de vinculação da barra não afetarão a resolução das equação:

$$\begin{cases} N(x) = \int q_x(x)dx \\ V(x) = \int q_y(x)dx \\ M(x) = \iint q_y(x)dx \end{cases} \tag{B.1}$$

Ademais, é importante que se tenha em mente que os esforços internos não seguirão a mesma convenção de sinais que as demais forças, mas a sua própria, também definida no quadro 2.

B.1 Esforço normal $N(x)$

Sabemos do que foi definido na seção A.1 que a equação B.1 relativa aos esforços normais poderá ser escrita em termos dos coeficientes da carga linearmente distribuída. Além disto, também sabemos do desenvolvimento do método dos deslocamento que os esforços normais nas extremidades da barra é definido, podendo portanto ser representados por N_0 e N_L. Deste modo, levando em conta a convenção de sinais:

$$\begin{aligned} N(x) = \int q_x(x)dx = -\int q_{x0}\left(1 - \frac{x}{L}\right) + q_{x1}\left(\frac{x}{L}\right)dx \\ N(x) = -q_{x0}\left(x - \frac{x^2}{2L}\right) - q_{x1}\left(\frac{x^2}{2L}\right) + \mathbb{C}_0 \\ N(x=0) = -N_0 \quad \therefore \quad \mathbb{C}_0 = -N_0 \\ N(x) = -N_0 - q_{x0}\left(x - \frac{x^2}{2L}\right) - q_{x1}\left(\frac{x^2}{2L}\right) \end{aligned}$$

B.2 Esforço Cortante $V(x)$

Para a curva do esforço cortante, utilizaremos as mesmas disposições dos esforços normais. Poderemos então escrever:

$$\begin{aligned} V(x) = \int q_y(x)dx = \int q_{y0}\left(1 - \frac{x}{L}\right) + q_{y1}\left(\frac{x}{L}\right)dx \\ V(x) = q_{y0}\left(x - \frac{x^2}{2L}\right) + q_{y1}\left(\frac{x^2}{2L}\right) + \mathbb{C}_1 \\ V(x=0) = V_0 \quad \therefore \quad \mathbb{C}_1 = V_0 \\ V(x) = V_0 + q_{y0}\left(x - \frac{x^2}{2L}\right) + q_{y1}\left(\frac{x^2}{2L}\right) \end{aligned}$$

B.3 Momento Fletor $M(x)$

As mesmas disposições poderão ser utilizadas na resolução desta equação, sendo necessária apenas uma ressalva primária ao cálculo: pelo fato de a equação conter uma integral dupla, serão necessárias duas condições de contorno para determinar o valor das constantes de integração. Sabendo que $\frac{dM(x)}{dx} = V(x)$, podemos escrever que $\frac{dM(x=0)}{dx} = -V_0$

$$\begin{aligned} M(x) = \iint q_y(x)dxdx = -\iint q_{y0}\left(1 - \frac{x}{L}\right) + q_{y1}\left(\frac{x}{L}\right)dxdx \\ M(x) = -q_{y0}\left(\frac{x^2}{2} - \frac{x^3}{6L}\right) - q_{y1}\left(\frac{x^3}{6L}\right) + \mathbb{C}_2 x + \mathbb{C}_3 \\ M(x=0) = M_0 \quad \therefore \quad \mathbb{C}_3 = M_0 \\ \frac{dM(x=0)}{dx} = -V_0 \quad \therefore \quad \mathbb{C}_2 = -V_0 \\ M(x) = M_0 - V_0 x - q_{y0}\left(\frac{x^2}{2} - \frac{x^3}{6L}\right) - q_{y1}\left(\frac{x^3}{6L}\right) \end{aligned}$$

B.4 Diagrama de deformações

Para determinar a equação da linha elástica da estrutura, a separaremos em duas componentes: a equação respectiva aos deslocamentos axiais u e a respectiva aos deslocamentos transversais v. Para a primeira, teremos uma aproximação de elemento de treliça, portanto um grau de liberdade e uma equação de reta. Já para a segunda, teremos uma aproximação de elementos de viga, ou seja, dois graus de liberdade e uma equação do terceiro grau.

B.4.1 Deformações u no eixo x

Os deslocamentos axiais podem ser escritos a partir de uma interpolação do primeiro grau. Como condicionadores, dispomos dos deslocamentos no dito eixo em ambas as extremidades da barra, isto é, d_0 e d_3. Assim:

$$\begin{aligned} u(x) &= \mathbb{C}_0 d_0 + \mathbb{C}_1 d_3 \\ u(0) = d_0;\quad u(L) = d_3 \quad \therefore \quad \mathbb{C}_0 &= \left(1 - \frac{x}{L}\right);\quad \mathbb{C}_1 = \left(\frac{x}{L}\right) \\ u(x) &= \left(1 - \frac{x}{L}\right) d_0 + \left(\frac{x}{L}\right) d_3 \end{aligned}$$

B.4.2 Deformações v no eixo y

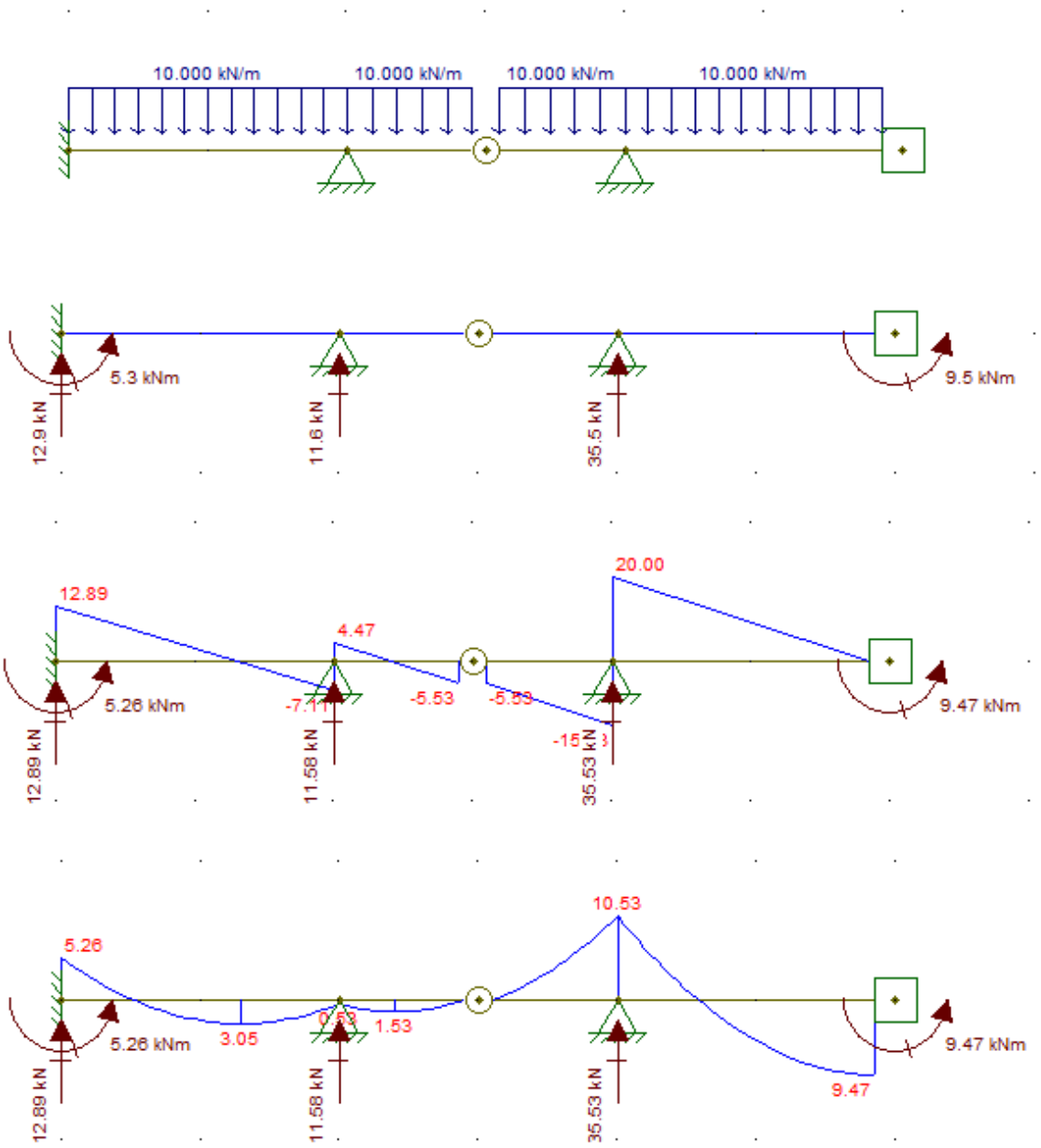
Já os deslocamentos transversais deverão ser interpolados por uma função do terceiro grau, o que implica na necessidade de quatro condições de contorno, para as quais utilizaremos os deslocamentos em y, d_1 e d_4 e as rotações em z, d_2 e d_5, na extremidade inicial e final da barra, respectivamente. Deste modo:

$$\begin{aligned} v(x) &= \mathbb{C}_0 d_1 + \mathbb{C}_1 d_4 + \mathbb{C}_2 d_2 + \mathbb{C}_3 d_5 \\ v(0) = d_1;\quad v(L) = d_4; \frac{\mathrm{d}v}{\mathrm{d}x}(0) = d_2;\quad \frac{\mathrm{d}v}{\mathrm{d}x}(L) &= d_5 \quad \therefore \\ \mathbb{C}_0 = 1 - 3\left(\frac{x}{L}\right)^2 + 2\left(\frac{x}{L}\right)^3;\quad \mathbb{C}_1 = 3\left(\frac{x}{L}\right)^2 - 2\left(\frac{x}{L}\right)^3;& \\ \mathbb{C}_2 = x\left(1 - \frac{x}{L}\right)^2;\quad \mathbb{C}_3 = x\left(\left(\frac{x}{L}\right)^2 - \frac{x}{L}\right);& \\ v(x) = \left(1 - 3\left(\frac{x}{L}\right)^2 + 2\left(\frac{x}{L}\right)^3\right) d_1 + \left(3\left(\frac{x}{L}\right)^2 - 2\left(\frac{x}{L}\right)^3\right) d_4 +& \\ \left(x\left(1 - \frac{x}{L}\right)^2\right) d_2 + \left(x\left(\left(\frac{x}{L}\right)^2 - \frac{x}{L}\right)\right) d_5& \end{aligned}$$

APÊNDICE C – Validação dos resultados

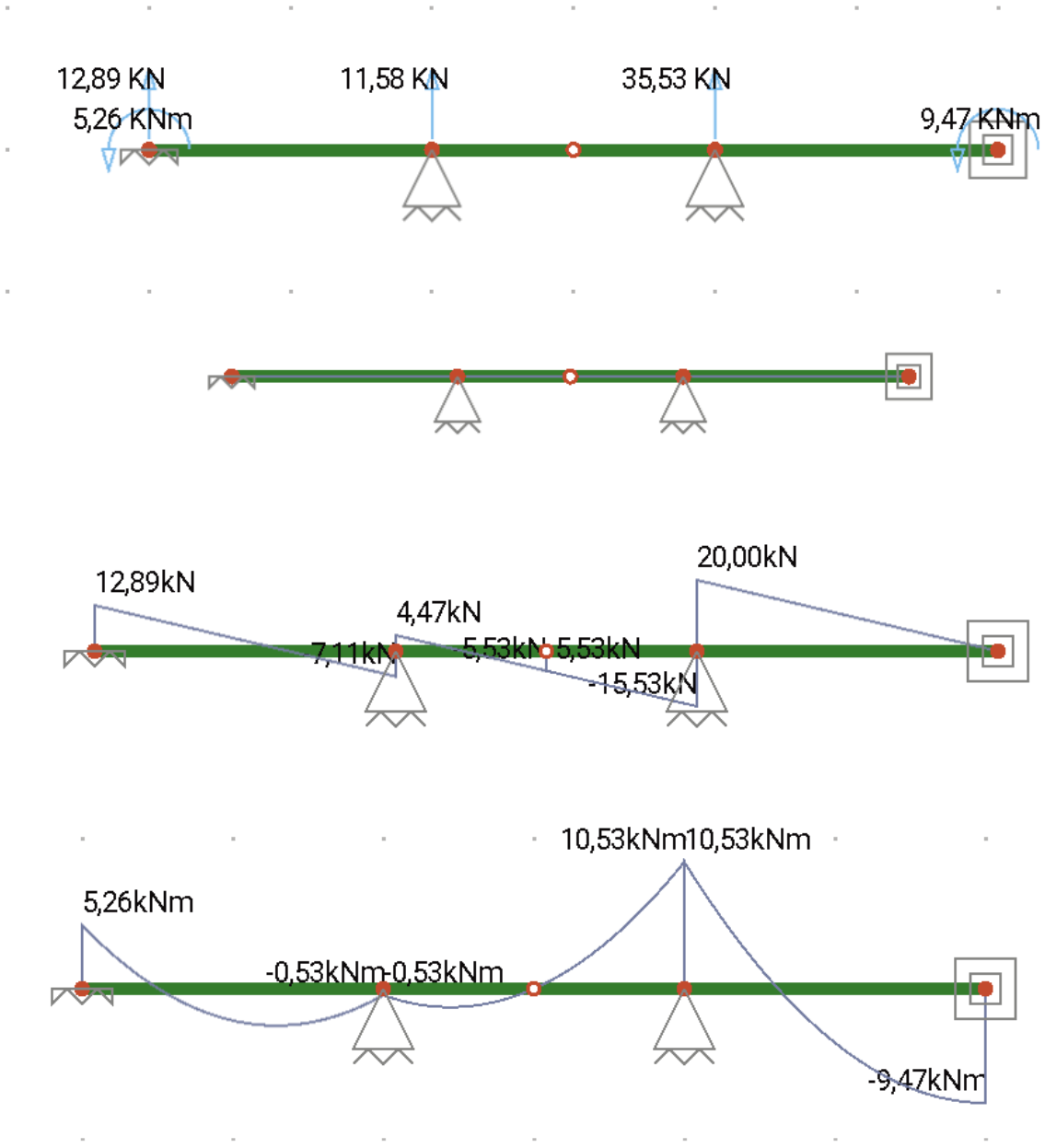
C.1 Viga Contínua

Figura 40 – Resultados apresentados pelo F-tool



Fonte: autor.

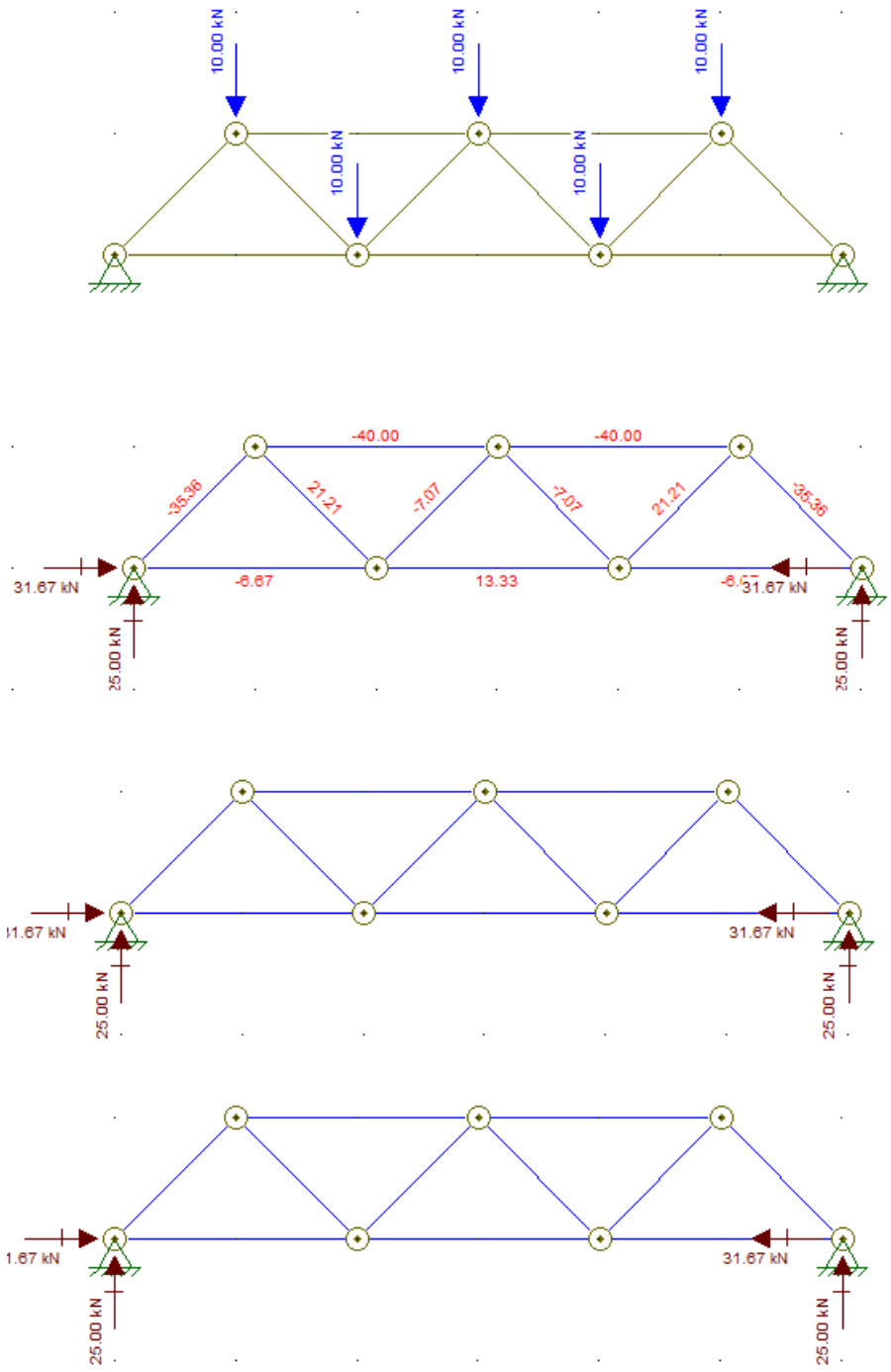
Figura 41 – Resultados apresentados pelo Aplicativo



Fonte: autor.

C.2 Treliça

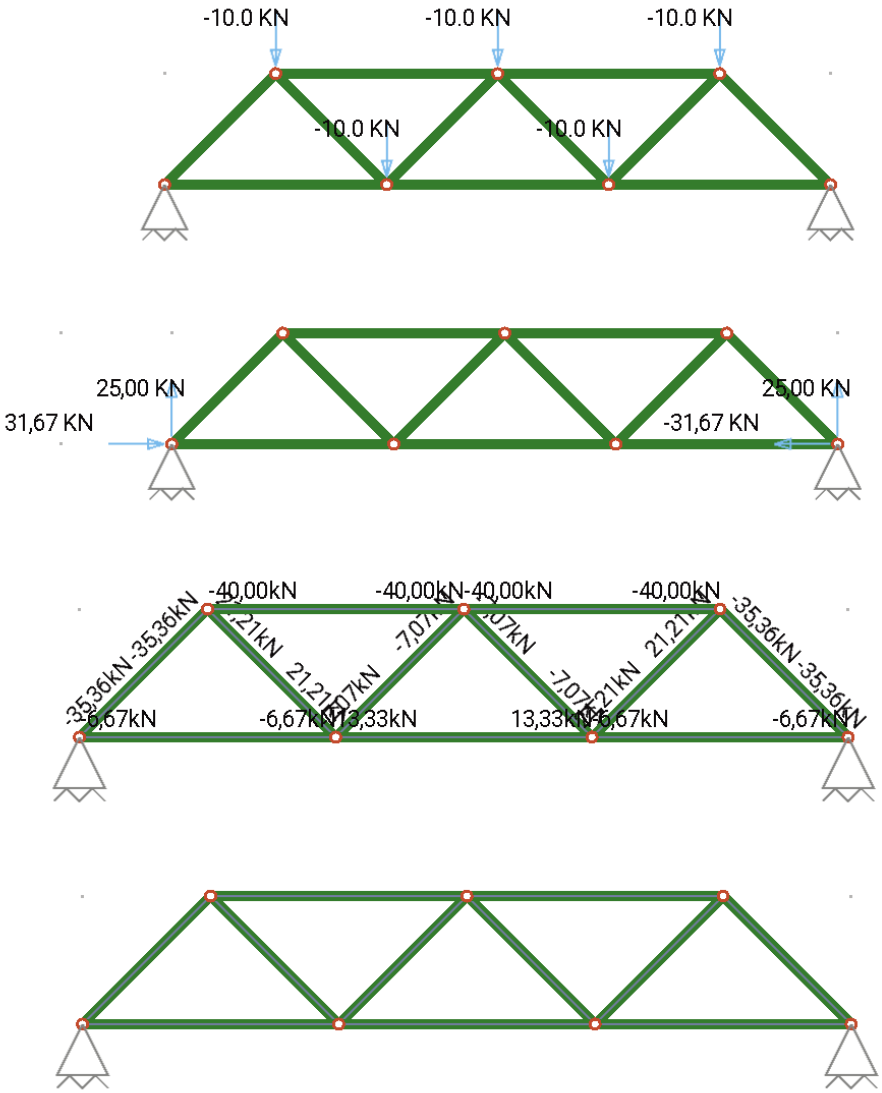
Figura 42 – Resultados apresentados pelo F-tool



Fonte: autor.

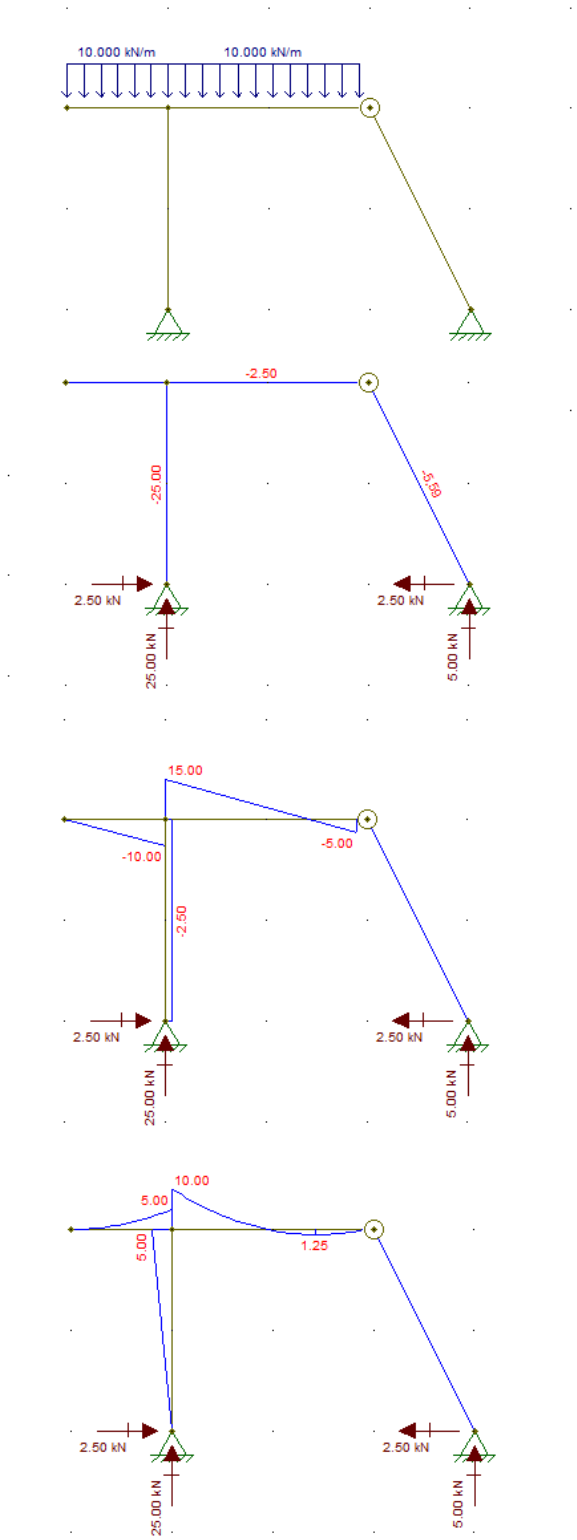
C.3 Pórtico

Figura 43 – Resultados apresentados pelo Aplicativo



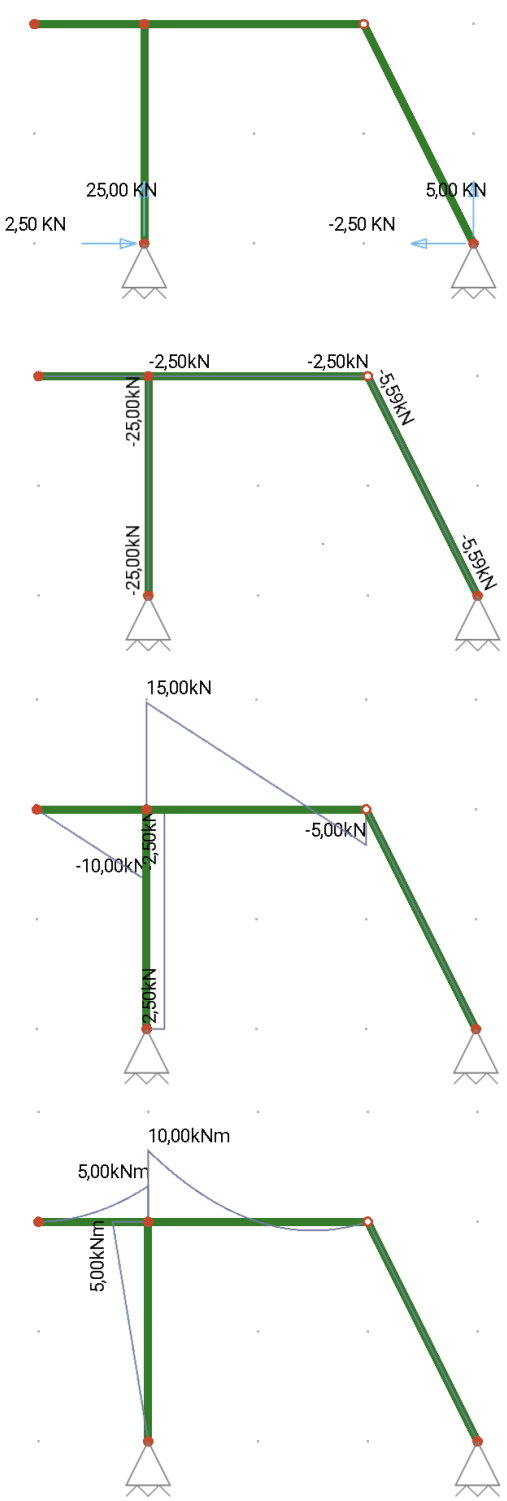
Fonte: autor.

Figura 44 – Resultados apresentados pelo Ftool



Fonte: autor.

Figura 45 – Resultados apresentados pelo Aplicativo



Fonte: autor.

APÊNDICE D – Código Java dos objetos

D.1 Classe Ponto.java

Ponto.java

```
1 package com.tcc.ecalc;
2
3 public class Ponto {
4
5     private double x, y;
6
7     //-------------------------------------------------------------Construtores
8     public Ponto(double x, double y) {
9         this.x = x;
10        this.y = y;
11    }
12
13    public Ponto(Ponto P) {
14        this.x = P.x;
15        this.y = P.y;
16    }
17
18    public Ponto() {
19        this.x = 0;
20        this.y = 0;
21    }
22
23    //-------------------------------------------------------------Getters
24
25    public double getX() {
26        return this.x;
27        }
28
29    public double getY() {
30        return this.y;
31        }
32
33    public double getMagnitude() {
34        return this.x*this.x+this.y*this.y;
35    }
36
37    //-------------------------------------------------------------Setters
38
39    public void setX(double x) {
40        this.x = x;
41        }
42
43    public void setY(double y) {
44        this.y = y;
45        }
46
47    public void setX(float x) {
48        this.x = x;
49        }
50
51    public void setY(float y) {
52        this.y = y;
53        }
54
55    //-------------------------------------------------------------Funções
56
57    public double getDistancia(Ponto B) {
58        return Math.sqrt(Math.pow(B.x-this.x, 2)+Math.pow(B.y-this.y, 2));
59    }
60
61    public static double getDistancia(Ponto A, Ponto B) {
```

```
62          return Math.sqrt(Math.pow(A.x-B.x, 2)+Math.pow(A.y-B.y, 2));
63      }
64
65      public double getAngulo(Ponto B) {
66          double a;
67          if(B.x != this.x) {
68              a = Math.atan((B.y-this.y)/(B.x-this.x));
69              a = Math.toDegrees(a);}
70          else {
71              if(B.y < this.y) {
72                  a = 90;
73              } else {
74                  a = -90;
75              }
76          }
77          return a;
78      }
79
80      public double getdX(Ponto P) {
81          return P.x-this.x;
82      }
83
84      public double getdY(Ponto P) {
85          return P.y-this.y;
86      }
87
88      public boolean isEqual(Ponto P) {
89          return ((this.x == P.x) && (this.y == P.y));
90      }
91
92      public static double maxX(Ponto[] P) {
93          double maxX = P[0].getX();
94          for(int i = 0; i < P.length; i++) {
95              if(P[i].getX()>maxX) {
96                  maxX = P[i].getX();
97              }
98          }
99          return maxX;
100     }
101
102     public static double minX(Ponto[] P) {
103         double minX = P[0].getX();
104         for(int i = 0; i < P.length; i++) {
105             if(P[i].getX()<minX) {
106                 minX = P[i].getX();
107             }
108         }
109         return minX;
110     }
111
112     public static double maxY(Ponto[] P) {
113         double maxY = P[0].getY();
114         for(int i = 0; i < P.length; i++) {
115             if(P[i].getY()>maxY) {
116                 maxY = P[i].getY();
117             }
118         }
119         return maxY;
120     }
121
122     public static double minY(Ponto[] P) {
```

```
123         double minY = P[0].getY();
124         for(int i = 0; i < P.length; i++) {
125             if(P[i].getY()<minY) {
126                 minY = P[i].getY();
127             }
128         }
129         return minY;
130     }
131 
132     public Ponto mult(double c) {
133         Ponto A = new Ponto(this);
134         A.x *= c;
135         A.y *= c;
136         return A;
137     }
138 
139     public Ponto mult(Ponto E) {
140         Ponto A = new Ponto(this);
141         A.x = A.x*E.x;
142         A.y = A.y*E.y;
143         return A;
144     }
145 
146     public Ponto div(Ponto E) {
147         Ponto A = new Ponto(this);
148         A.x = A.x/E.x;
149         A.y = A.y/E.y;
150         return A;
151     }
152 
153     public Ponto div(double c) {
154         Ponto A = new Ponto(this);
155         A.x = A.x/c;
156         A.y = A.y/c;
157         return A;
158     }
159 
160     public Ponto soma(Ponto X) {
161         Ponto R = new Ponto(0,0);
162         R.setX(this.getX()+X.getX());
163         R.setY(this.getY()+X.getY());
164         return R;
165     }
166 
167     public Ponto subtrai(Ponto X) {
168         Ponto R = new Ponto(0,0);
169         R.setX(this.getX()-X.getX());
170         R.setY(this.getY()-X.getY());
171         return R;
172     }
173 
174     public static Ponto origem() {
175         return new Ponto(0,0);
176     }
177 
178     public String toString() {
179         return "("+this.x+","+this.y+")";
180     }
181 
182     //Determina se quatro pontos formam duas retas coincidentes
183     public static boolean coinc(Ponto A1, Ponto A2, Ponto B1, Ponto B2) {
```

```
184         double a1 = A1.y - A2.y;
185         double b1 = A2.x - A1.x;
186         double c1 = A1.x*A2.y-A1.y*A2.x;
187         System.out.println(a1+"*x + "+b1+"*y + "+c1+"=0");
188         double a2 = B1.y - B2.y;
189         double b2 = B2.x - B1.x;
190         double c2 = B1.x*B2.y-B1.y*B2.x;
191         System.out.println(a2+"*x + "+b2+"*y + "+c2+"=0");
192         if(a1 != 0 && a2 != 0) {
193             System.out.println("Dividindo por a = "+a1+", "+a2);
194             b1 /= a1;
195             c1 /= a1;
196             a1 /= a1;
197             b2 /= a2;
198             c2 /= a2;
199             a2 /= a2;
200         } else {
201             if(b1 != 0 && b2 != 0) {
202                 System.out.println("Dividindo por b = "+b1+", "+b2);
203                 a1 /= b1;
204                 c1 /= b1;
205                 b1 /= b1;
206                 a2 /= b2;
207                 c2 /= b2;
208                 b2 /= b2;
209             } else {
210                 if(c1 != 0 && c2 != 0) {
211                     System.out.println("Dividindo por c = "+c1+", "+c2);
212                     a1 /= c1;
213                     b1 /= c1;
214                     c1 /= c1;
215                     a2 /= c2;
216                     b2 /= c2;
217                     c2 /= c2;
218                 }
219             }
220         }
221         System.out.println(a1+"*x + "+b1+"*y + "+c1+"=0");
222         System.out.println(a2+"*x + "+b2+"*y + "+c2+"=0");
223         if((a1 == a2 && b1 == b2 && c1 == c2) &&
224            (A2.soma(A1).div(2).getDistancia(B2.soma(B1).div(2)) <
    A2.getDistancia(A1)/2+B2.getDistancia(B1)/2)){
225             return true;
226         } else {
227             return false;
228         }
229     }
230 
231     public static double distReta(Ponto P1, Ponto P2, Ponto P) {
232         double a = P1.getY() - P2.getY();
233         double b = P2.getX() - P1.getX();
234         double c = P1.getX()*P2.getY()-P1.getY()*P2.getX();
235         double d = Math.abs(a*P.getX()+b*P.getY()+c)/Math.sqrt(a*a+b*b);
236         return d;
237     }
238 
239     //Determina o ponto de cruzamento C que quatro pontos formam
240     public static Ponto cruzamento(Ponto A1, Ponto A2, Ponto B1, Ponto B2, Ponto C) {
241         double a1 = A1.y - A2.y;
242         double b1 = A2.x - A1.x;
243         double c1 = A1.x*A2.y-A1.y*A2.x;
```

```
244         double a2 = B1.y - B2.y;
245         double b2 = B2.x - B1.x;
246         double c2 = B1.x*B2.y-B1.y*B2.x;
247         double x = (b1*c2-b2*c1)/(a1*b2-a2*b1);
248         double y = (a2*c1-a1*c2)/(a1*b2-a2*b1);
249         Ponto Cruz = new Ponto(x,y);
250         if(((A2.soma(A1).div(2)).getDistancia(Cruz) <= A1.getDistancia(A2)/2) &&
251            ((B2.soma(B1).div(2)).getDistancia(Cruz) <= B1.getDistancia(B2)/2)) {
252             return Cruz;
253         }
254         return C;
255     }
256 }
257
```

D.2 Classe BigMatrix.java

BigMatrix.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.io.OutputStreamWriter;
10
11 public class BigMatrix {
12
13     public int m, n;
14     public BigDecimal[][] dados;
15     public int p;
16     public static DecimalFormat f = new DecimalFormat("0.#E+00");
17
18     public BigMatrix(int m, int n, int p) {
19         this.m = m;
20         this.n = n;
21         this.p = p;
22         this.dados = new BigDecimal[m][n];
23     }
24
25     public BigMatrix(int m, int n, int p, boolean isIni) {
26         this.m = m;
27         this.n = n;
28         this.p = p;
29         this.dados = new BigDecimal[m][n];
30         if(isIni) {
31             for(int i = 0; i < m; i++) {
32                 for(int j = 0; j < n; j++) {
33                     this.dados[i][j] = new BigDecimal("0");
34                 }
35             }
36         }
37     }
38
39     public BigMatrix(BigDecimal[][] dados, int p) {
40         this.m = dados.length;
41         this.n = dados[0].length;
42         this.p = p;
43         for(int i = 0; i < m; i++) {
44             for(int j = 0; j < n; j++) {
45                 this.dados[i][j] = dados[i][j];
46             }
47         }
48     }
49
50     public static BigMatrix rigidez(BigDecimal E, BigDecimal I, BigDecimal A, BigDecimal L,
   int mt, int p) {
51         BigMatrix R = new BigMatrix(6, 6, p, true);
52         BigDecimal EAL = E.multiply(A).divide(L, p, RoundingMode.HALF_EVEN);
53         BigDecimal EIL = E.multiply(I).divide(L, p, RoundingMode.HALF_EVEN);
54         BigDecimal EIL2 = E.multiply(I).divide(L.pow(2), p, RoundingMode.HALF_EVEN);
55         BigDecimal EIL3 = E.multiply(I).divide(L.pow(3), p, RoundingMode.HALF_EVEN);
56         BigDecimal n1 = new BigDecimal("-1");
57         Log.d("Rigidez",EAL+" "+EIL+" "+EIL2+" "+EIL3);
58         R.set(0, 0,EAL); R.set(0, 3, EAL.multiply(n1));
59         R.set(3, 3,EAL); R.set(3, 0, EAL.multiply(n1));
60         switch(mt) {
61         case 0 : {
62             R.set(1, 1, EIL3.multiply(new BigDecimal("12")));
```

```
63              R.set(1, 2, EIL2.multiply(new BigDecimal("6")));
64              R.set(1, 4, EIL3.multiply(new BigDecimal("-12")));
65              R.set(1, 5, EIL2.multiply(new BigDecimal("6")));
66              R.set(2, 1, EIL2.multiply(new BigDecimal("6")));
67              R.set(2, 2, EIL .multiply(new BigDecimal("4")));
68              R.set(2, 4, EIL2.multiply(new BigDecimal("-6")));
69              R.set(2, 5, EIL .multiply(new BigDecimal("2")));
70              R.set(4, 1, EIL3.multiply(new BigDecimal("-12")));
71              R.set(4, 2, EIL2.multiply(new BigDecimal("-6")));
72              R.set(4, 4, EIL3.multiply(new BigDecimal("12")));
73              R.set(4, 5, EIL2.multiply(new BigDecimal("-6")));
74              R.set(5, 1, EIL2.multiply(new BigDecimal("6")));
75              R.set(5, 2, EIL .multiply(new BigDecimal("2")));
76              R.set(5, 4, EIL2.multiply(new BigDecimal("-6")));
77              R.set(5, 5, EIL .multiply(new BigDecimal("4")));
78              break;
79          }
80          case 1 : {
81              R.set(1, 1, EIL3.multiply(new BigDecimal("3")));
82              R.set(1, 4, EIL3.multiply(new BigDecimal("-3")));
83              R.set(1, 5, EIL2.multiply(new BigDecimal("3")));
84              R.set(4, 1, EIL3.multiply(new BigDecimal("-3")));
85              R.set(4, 4, EIL3.multiply(new BigDecimal("3")));
86              R.set(4, 5, EIL2.multiply(new BigDecimal("-3")));
87              R.set(5, 1, EIL2.multiply(new BigDecimal("3")));
88              R.set(5, 4, EIL2.multiply(new BigDecimal("-3")));
89              R.set(5, 5, EIL .multiply(new BigDecimal("3")));
90              break;
91          }
92          case 2 : {
93              R.set(1, 1, EIL3.multiply(new BigDecimal("3")));
94              R.set(1, 2, EIL2.multiply(new BigDecimal("3")));
95              R.set(1, 4, EIL3.multiply(new BigDecimal("-3")));
96              R.set(2, 1, EIL2.multiply(new BigDecimal("3")));
97              R.set(2, 2, EIL .multiply(new BigDecimal("3")));
98              R.set(2, 4, EIL2.multiply(new BigDecimal("-3")));
99              R.set(4, 1, EIL3.multiply(new BigDecimal("-3")));
100             R.set(4, 2, EIL2.multiply(new BigDecimal("-3")));
101             R.set(4, 4, EIL3.multiply(new BigDecimal("3")));
102             break;
103         }
104         }
105         return R;
106     }
107
108     public static BigMatrix rotacao(BigDecimal a,int p) {
109         BigDecimal sin = new BigDecimal(Math.sin(Math.toRadians(a.doubleValue())));
110         BigDecimal cos = new BigDecimal(Math.cos(Math.toRadians(a.doubleValue())));
111         BigMatrix r = new BigMatrix(6,6,p,true);
112         r.set(0,0,cos);
113         r.set(1,0,sin.multiply(new BigDecimal("-1")));
114         r.set(0,1,sin);
115         r.set(1,1,cos);
116         r.set(2, 2, new BigDecimal("1"));
117         r.set(3,3,cos);
118         r.set(4,3,sin.multiply(new BigDecimal("-1")));
119         r.set(3,4,sin);
```

```
120         r.set(4,4,cos);
121         r.set(5,5, new BigDecimal("1"));
122         return r;
123     }
124 
125     public static BigMatrix engPerfeit(CargaDistribuida Q, BigDecimal L, int mt, int p) {
126         BigMatrix eP = new BigMatrix(6,1,p);
127         BigDecimal xA = new BigDecimal(Q.getQ(0));
128         BigDecimal yA = new BigDecimal(Q.getQ(1));
129         BigDecimal xB = new BigDecimal(Q.getQ(2));
130         BigDecimal yB = new BigDecimal(Q.getQ(3));
131         Log.d("EngPerf",xA + " " + yA + " " + xB + " " + yB);
132         eP.set(0, 0, xA.multiply(new BigDecimal(2)).add(xB).multiply(L)
133                         .divide(new BigDecimal("-6"), p, RoundingMode.HALF_EVEN));
134         eP.set(3, 0, xA.add(xB.multiply(new BigDecimal(2))).multiply(L)
135                         .divide(new BigDecimal("-6"), p, RoundingMode.HALF_EVEN));
136         switch(mt) {
137         case 0 : {
138             eP.set(1, 0, yA.multiply(new BigDecimal("7")).add(yB.multiply(new
    BigDecimal("3")))
139                         .multiply(L).divide(new BigDecimal("-20"), p, RoundingMode.HALF_EVEN));
140             eP.set(4, 0, yA.multiply(new BigDecimal("3")).add(yB.multiply(new
    BigDecimal("7")))
141                         .multiply(L).divide(new BigDecimal("-20"), p, RoundingMode.HALF_EVEN));
142             eP.set(2, 0, (yA.multiply(new BigDecimal("6")) .add(yB.multiply(new
    BigDecimal("4"))))
143                         .multiply(L.pow(2)).divide(new BigDecimal("-120"), p,
    RoundingMode.HALF_EVEN));
144             eP.set(5, 0, yA.multiply(new BigDecimal("4")).add(yB.multiply(new
    BigDecimal("6")))
145                         .multiply(L.pow(2)).divide(new BigDecimal("120"), p,
    RoundingMode.HALF_EVEN));
146             break;
147         }
148         case 1 : {
149             eP.set(1, 0, yA.multiply(new BigDecimal("33")).add(yB.multiply(new
    BigDecimal("12")))
150                         .multiply(L).divide(new BigDecimal("-120"), p,
    RoundingMode.HALF_EVEN));
151             eP.set(4, 0, yA.multiply(new BigDecimal("27")).add(yB.multiply(new
    BigDecimal("48")))
152                     .multiply(L).divide(new BigDecimal("-120"), p, RoundingMode.HALF_EVEN));
153             eP.set(2, 0, new BigDecimal("0"));
154             eP.set(5, 0, yA.multiply(new BigDecimal("7")).add(yB.multiply(new
    BigDecimal("8")))
155                     .multiply(L.pow(2)).divide(new BigDecimal("120"), p,
    RoundingMode.HALF_EVEN));
156             break;
157         }
158         case 2 : {
159             eP.set(1, 0, yA.multiply(new BigDecimal("48")).add(yB.multiply(new
    BigDecimal("27")))
160                         .multiply(L).divide(new BigDecimal("-120"), p,
    RoundingMode.HALF_EVEN));
161             eP.set(4, 0, yA.multiply(new BigDecimal("12")).add(yB.multiply(new
    BigDecimal("33")))
162                         .multiply(L).divide(new BigDecimal("-120"), p,
```

```
   RoundingMode.HALF_EVEN));
163             eP.set(2, 0, yA.multiply(new BigDecimal("8")) .add(yB.multiply(new
   BigDecimal("7")))
164                     .multiply(L.pow(2)).divide(new BigDecimal("-120"), p,
   RoundingMode.HALF_EVEN));
165             eP.set(5, 0, new BigDecimal("0"));
166             break;
167         }
168         case 3 : {
169             eP.set(1, 0, yA.multiply(new BigDecimal("2")).add(yB)
170                     .multiply(L).divide(new BigDecimal("-6"), p, RoundingMode.HALF_EVEN));
171             eP.set(4, 0, yA.add(yB.multiply(new BigDecimal("2")))
172                     .multiply(L).divide(new BigDecimal("-6"), p, RoundingMode.HALF_EVEN));
173             eP.set(2, 0, new BigDecimal("0"));
174             eP.set(5, 0, new BigDecimal("0"));
175             break;
176         }
177         }
178         return eP;
179     }
180
181
182     public void set(int i, int j, BigDecimal v) {
183         dados[i][j] = v;
184     }
185
186     public void setScale(int p) {
187         for(int i = 0; i < this.m; i++) {
188             for(int j = 0; j < this.n; j++) {
189                 this.dados[i][j].setScale(p,RoundingMode.HALF_EVEN);
190             }
191         }
192     }
193
194     public BigDecimal get(int i, int j) {
195         return dados[i][j];
196     }
197
198     public BigMatrix multLinha(int i, BigDecimal c) {
199         for(int j = 0; j < this.n; j++) {
200             dados[i][j] = dados[i][j].multiply(c);
201         }
202         return this;
203     }
204
205     public BigMatrix swapLinha(int i, int k) {
206         BigDecimal aux;
207         for(int j = 0; j < n; j++) {
208             aux = dados[i][j];
209             dados[i][j] = dados[k][j];
210             dados[k][j] = aux;
211         }
212         return this;
213     }
214
215     public BigMatrix transpoe() {
216         BigMatrix C = new BigMatrix(this.n, this.m, this.p);
```

```
217         for(int i = 0; i < this.m; i++) {
218             for(int j = 0; j < this.n; j++) {
219                 C.dados[i][j] = this.dados[j][i];
220             }
221         }
222         return C;
223     }
224 
225     public BigMatrix mult(BigMatrix B) {
226         BigMatrix C = new BigMatrix(this.m, B.n, Math.min(this.p, B.p), true);
227         for(int i = 0; i < m; i++) {
228             for(int j = 0; j < B.n; j++) {
229                 for(int k = 0; k < n; k++) {
230                     C.somaCelula(i, j, this.dados[i][k].multiply(B.dados[k][j]));
231                 }
232             }
233         }
234         return C;
235     }
236 
237 
238 
239     public BigMatrix linhaDivC(int i, BigDecimal C) {
240         for(int j = 0; j < n; j++) {
241             dados[i][j] = dados[i][j].divide(C, p, RoundingMode.HALF_EVEN);
242         }
243         return this;
244     }
245 
246     public BigMatrix divC(BigDecimal C) {
247         for(int i = 0; i < m; i++) {
248             for(int j = 0; j < n; j++) {
249                 dados[i][j] = dados[i][j].divide(C, p, RoundingMode.HALF_EVEN);
250             }
251         }
252         return this;
253     }
254 
255     public BigMatrix linhaSomaMultC(int i, int k, BigDecimal C) {
256         for(int j = 0; j < this.n; j++) {
257             this.dados[i][j] = this.dados[i][j].add(this.dados[k][j].multiply(C));
258         }
259         return this;
260     }
261 
262     public BigMatrix somaCelula(int i, int j, BigDecimal v) {
263         this.dados[i][j] = this.dados[i][j].add(v);
264         return this;
265     }
266 
267     public BigMatrix soma(BigMatrix C) {
268         for(int i = 0; i < this.m; i++) {
269             for(int j = 0; j < this.n; j++) {
270                 this.dados[i][j] = this.dados[i][j].add(C.dados[i][j]);
271             }
272         }
273         return this;
```

```
274     }
275
276     public BigMatrix subtrai(BigMatrix C) {
277         for(int i = 0; i < this.m; i++) {
278             for(int j = 0; j < this.n; j++) {
279                 this.dados[i][j] = this.dados[i][j].subtract(C.dados[i][j]);
280             }
281         }
282         return this;
283     }
284
285     @Override
286     public String toString() {
287         f.setMinimumFractionDigits(15);
288         f.setMaximumFractionDigits(15);
289         f.setPositivePrefix(" ");
290         f.setNegativePrefix("-");
291         String matrix = new String();
292         for(int i = 0; i < m; i++) {
293             matrix += "[";
294             for(int j = 0; j < n; j++) {
295                 matrix += f.format(dados[i][j].doubleValue())+" ";
296             }
297             matrix += "]\n";
298         }
299         return matrix;
300     }
301
302     public static BigMatrix resolve(BigMatrix A, BigMatrix B, Context ctx) throws Exception {
303         BigMatrix S = new BigMatrix(A.m, A.n+1, Math.min(A.p,B.p));
304         for(int i = 0; i < A.m; i++) {
305             for(int j = 0; j < A.n; j++) {
306                 S.set(i, j, A.dados[i][j]);
307             }
308             S.set(i, S.n-1, B.dados[i][0]);
309         }
310         return resolve(S, ctx);
311     }
312
313     public static BigMatrix resolve(BigMatrix S, Context ctx) throws Exception {
314         OutputStreamWriter mWriter;
315             mWriter = new OutputStreamWriter(ctx.openFileOutput("solve.txt",
    Context.MODE_PRIVATE));
316             mWriter.write(S.toString()+"\n\n");
317         for(int i = 0; i < S.m; i++) {
318             Log.d("Solução", "Linha "+i);
319             int count = 1;
320             while(S.dados[i][i].compareTo(BigDecimal.ZERO) == 0) {
321                 if(i+count < S.m) {
322                     S.swapLinha(i, count);
323                     count += 1;
324                     Log.d("Solução","Troca de linhas"+i+" "+(i+count));
325                     } else {
326                         mWriter.flush();
327                         throw new
    Exception(ctx.getResources().getString(R.string.hipostaticaexception));
328                 }
```

```
329             }
330             mWriter.write("Divide matriz por" + f.format(S.dados[i][i]) +"\n");
331             S = S.linhaDivC(i, S.dados[i][i]);
332             mWriter.write(S.toString()+"\n\n");
333 
334             for(int k = 0; k < S.m; k++) {
335                 if(k != i && S.dados[k][i].compareTo(new BigDecimal("0")) != 0) {
336                     mWriter.write("L"+k+"<- L"+k+" - "+f.format(S.dados[k][i])+"L"+i+"\n");
337                     S = S.linhaSomaMultC(k, i, S.dados[k][i].multiply(new BigDecimal(-1)));
338                     mWriter.write(S.toString()+"\n\n");
339                 }
340             }
341         }
342 
343         BigMatrix R = new BigMatrix(S.m,1,S.p);
344         for(int i = 0; i < S.m; i++) {
345             R.set(i, 0, S.get(i, S.n-1));
346         }
347 
348         mWriter.flush();
349         return R;
350     }
351 }
352 
353 
```

D.3 Classe CargaPontual.java

CargaPontual.java

```java
package com.tcc.ecalc;

public class CargaPontual {
    private int _id; //Id
    private String nome; //Nome
    private double[] P = new double[3]; //Valores

    //Construtor de objeto
    public CargaPontual(int _id, String nome, double Px, double Py, double Pz) {
        this._id = _id;
        this.nome = nome;
        this.P[0] = Px;
        this.P[1] = Py;
        this.P[2] = Pz;
    }

    //Retorna uma string inteligível descrevendo a carga
    public String toString() {
        return "id = "+this._id+", nome = "+this.nome+", Px = "+this.P[0]+
               "Py = "+this.P[1]+", Pz = "+this.P[2];
    }

    //-----------------------------------------------------------------------Getters

    public int getId() {
        return this._id;
        }

    public String getNome() {
        return this.nome;
        }

    public double getP(int i) {
        return this.P[i];
        }

    public double[] getP() {
        return this.P;
        }

    //-----------------------------------------------------------------------Setters

    public void setNome(String nome) {
        this.nome = nome;
        }

    public void setP(int i, double P) {
        this.P[i] = P;
        }
}

```

D.4 Classe CargaDistribuida.java

CargaDistribuida.java

```java
package com.tcc.ecalc;

public class CargaDistribuida {
    private int _id; //Id da Carga Distribuída
    private String nome; //Nome
    private double[] Q = new double[4]; //Array com os valores no início e fim
    private boolean isGlobal; //Sistema

    //---------------------------------------------------------Construtores

    public CargaDistribuida(int _id, String nome, double Qx0, double Qy0, double Qx1, double Qy1, boolean isGlobal) {
        this._id = _id;
        this.nome = nome;
        this.Q[0] = Qx0;
        this.Q[1] = Qy0;
        this.Q[2] = Qx1;
        this.Q[3] = Qy1;
        this.isGlobal = isGlobal;
    }

    public CargaDistribuida(CargaDistribuida Q) {
        this._id = Q._id;
        this.nome = Q.nome;
        this.Q[0] = Q.Q[0];
        this.Q[1] = Q.Q[1];
        this.Q[2] = Q.Q[2];
        this.Q[3] = Q.Q[3];
        this.isGlobal = Q.isGlobal;
    }

    @Override //Retorna uma string inteligível descrevendo a carga distribuida
    public String toString() {
        return "id = "+this._id+
                ", nome = "+this.nome +
                ", Qx0 = "+this.Q[0] +
                ", Qy0 = "+this.Q[1] +
                ", Qx1 = "+this.Q[2] +
                ", Qy1 = "+this.Q[3] +
                ", isGlobal = "+this.isGlobal;
    }

    //-------------------------------------------Getters
    public int getID() {
        return this._id;
        }

    public String getNome() {
        return this.nome;
        }

    public double getQ(int i) {
        return this.Q[i];
        }

    public double[] getQ() {
        return this.Q;
        }

    public boolean isGlobal() {
        return this.isGlobal;
```

```
61          }
62
63      public double getMagnitude(int i) {
64          double mag;
65          mag = Math.sqrt(Q[i*2]*Q[i*2]+Q[i*2+1]*Q[i*2+1]);
66          return mag;
67      }
68
69      //-----------------------------------------------Setters
70
71      public void setNome(String nome) {
72          this.nome = nome;
73          }
74
75      public void setQ(int i, double Q) {
76          this.Q[i] = Q;
77          }
78
79      public void isGlobal(boolean isGlobal) {
80          this.isGlobal = isGlobal;
81          }
82
83  }
84
```

D.5 Classe Material.java

Material.java

```java
package com.tcc.ecalc;

public class Material {
    private int _id; //Id
    private String nome; //nome
    private double modE, //Módulo de elasticidade
                   dVol; //Densidade Volumétrica

    //COnstrutor de material
    public Material(int _id,String nome,double modE,double dVol) {
        this._id = _id;
        this.nome = nome;
        this.modE = modE;
        this.dVol = dVol;
    }

    //-------------------------------------------------------Getters

    public int getId() {
        return this._id;
        }

    public String getNome() {
        return this.nome;
        }

    public double getModE() {
        return this.modE;
        }

    public double getDVol() {
        return this.dVol;
        }
}

```

D.6 Classe Perfil.java

Perfil.java

```java
package com.tcc.ecalc;

import java.math.BigDecimal;

/*Devido a variedade de Perfis presentes na construção civil, é interessante que
* insiramos os mais comuns no programa. Isso auxiliará o usuário na entrada de
* dimensões, visto que não precisará calcular a área ou o momento de inércia des-
* tas formas - o programa o fará. Para identificar o tipo da seção, usaremos a
* variável tipo, de valores inteiros, que seguirá esta lista:
*
*   0 - Genérica: o própio usuário entra com os dados
*   1 - Circular;
*   2 - Retangular;
*   3 - Circular vazada;
*   4 - Retangular vazada;
*   5 - Tê;
*   6 - I;
*
*   //Dimensões que devem ser entradas na array
*  Dim.   |  b  |  h  | t,tw |  tf  |
*  D[i]   |  0  |  1  |  2   |  3   |
*/
//TODO Instituir constantes para ficar mais organizado
public class Perfil {

    private final static int p = 30;
    private int _id, //id
                tipo; //tipo
    private BigDecimal aSec, //Área da seção
                       momI; //Momento de inércia
    private final BigDecimal[] dim = new BigDecimal[4]; //Array de dimensões
    private String nome; //Nome do perfil

    //Construtor
    public Perfil(int _id, String nome, int tipo, BigDecimal[] dim) {
        this._id = _id;
        this.tipo = tipo;
        for(int i = 0; i < dim.length; i++) {
            this.dim[i] = dim[i];
        }
        this.aSec = this.calcArea();
        this.momI = this.calcInercia();
        this.nome = nome;
    }

    public Perfil(int _id, String nome, int tipo, BigDecimal aSec, BigDecimal momI,
  BigDecimal dim0, BigDecimal dim1, BigDecimal dim2, BigDecimal dim3) {
        this._id = _id;
        this.tipo = tipo;
        this.aSec = aSec;
        this.momI = momI;
        this.dim[0] = dim0;
        this.dim[1] = dim1;
        this.dim[2] = dim2;
        this.dim[3] = dim3;
        this.nome = nome;
    }


    //-----------------------------------------------------------------------Getters

```

```java
    @Override
    public String toString() {
        return "_id = "+_id+", nome = "+nome+", tipo = "+tipo+", aSec = "+aSec+", momI = "+momI+", dim("+dim[0]+"x"+dim[1]+"x"+dim[2]+"x"+dim[3]+")";
    }

    public int getId() {
        return this._id;
        }

    public int getTipo() {
        return this.tipo;
        }

    public String getNome() {
        return this.nome;
        }

    public BigDecimal getDim(int i) {
        return this.dim[i];
        }

    public BigDecimal[] getDim() {
        return this.dim;
        }

    public BigDecimal getASec() {
        return this.aSec;
        }

    public BigDecimal getMomI() {
        return this.momI;
        }

    //-----------------------------------------------------------------Setters
    public void setTipo(int i) {
        this.tipo = i;
        }

    public void setNome(String nome) {
        this.nome = nome;
        }

    public void setDim(int i, BigDecimal value) {
        this.dim[i] = value;
        }

    public void setASec(BigDecimal A) {
        this.aSec = A;
        }

    public void setMomI(BigDecimal I) {
        this.momI = I;
        }

    //-----------------------------------------------------------------Funções

    public BigDecimal calcArea() {

        BigDecimal A = BigDecimal.ZERO;
        switch (this.tipo) {
```

```
122             //Caso 0: Seção genérica.
123             case 1 : {
124                 A = dim[0];
125             }; break;
126             //Caso 1: Seção circular. A = (pi/4)*d^2
127             case 2 : {
128                 A = this.dim[0].multiply(this.dim[0]).divide(new
    BigDecimal("4").multiply(new BigDecimal(Math.PI)), p,RoundingMode.HALF_EVEN);
129             }; break;
130             //Caso 2: Seção retangular. A = b*h
131             case 3 : {
132                 A = this.dim[0].multiply(this.dim[1]);
133             }; break;
134             //Caso 3: Seção circular vazada. A = (PI/4)*(d1^2-(d1-2t)^2)
135             case 4 : {
136                 A =
    this.dim[0].pow(2).subtract((this.dim[0].subtract(this.dim[1].multiply(new
    BigDecimal(2)).pow(2)))).
137                         multiply(new BigDecimal(Math.PI/4));
138             }; break;
139             //Caso 4: Seção retangular vazada. A = b*h-(b-2t_w)*(h-2t_f)
140             case 5 : {
141                 BigDecimal a = this.dim[1].subtract(this.dim[3].multiply(new
    BigDecimal("2")));
142                 BigDecimal b = this.dim[0].subtract(this.dim[2].multiply(new
    BigDecimal("2")));
143                 BigDecimal c = this.dim[0].multiply(this.dim[1]);
144                 A = c.subtract(b.multiply(a));
145             }; break;
146             //Caso 5: Seção em Tê. A = b*tf+(h-tf)*tw;
147             case 6 : {
148                 BigDecimal a = this.dim[0].multiply(this.dim[3]);
149                 BigDecimal b = this.dim[1].subtract(this.dim[3]).multiply(this.dim[2]);
150                 A = a.add(b);
151             }; break;
152             //Caso 6: Seção em I. A=h*tw+2(b-tw)*tf;
153             case 7 : {
154                 BigDecimal a = this.dim[1].multiply(this.dim[2]);
155                 BigDecimal b =
    this.dim[0].subtract(this.dim[2]).multiply(this.dim[3]).multiply(new BigDecimal("2"));
156                 A = a.add(b);
157             }
158         }
159         return A;
160     }
161
162     public BigDecimal calcInercia() {
163             BigDecimal I = BigDecimal.ZERO;
164             switch (tipo) {
165                 //Caso 0: Seção genérica.
166                 case 1 : {
167                     I = this.dim[0];
168                 }; break;
169             //Caso 1: Seção circular. I = (pi/32)*d^4
170                 case 2 : {
171                     I = this.dim[0].pow(4).divide(new BigDecimal(Math.PI/64));
172                 }; break;
173                 //Caso 2: Seção retangular. I = 1/12 * b*h^3
174                 case 3 : {
175                     I = this.dim[0].multiply(this.dim[1].pow(3)).divide(new
    BigDecimal(12),p,RoundingMode.HALF_EVEN);
```

```
176                 }; break;
177                 //Caso 3: Seção circular vazada. I = (pi/32)*(d^4-(d-t)^4)
178                 case 4 : {
179                     BigDecimal a = this.dim[0].pow(4);
180                     BigDecimal b = this.dim[0].subtract(this.dim[1].multiply(new
    BigDecimal(2)).pow(4));
181                     I = a.subtract(b).multiply(new BigDecimal(Math.PI/64));
182                 }; break;
183                 //Caso 4: Seção retangular vazada. I = (1/12)*(b*h^3 - (b-2t_w)(h-2t_f)^3);
184                 case 5 : {
185                     BigDecimal a = this.dim[0].multiply(this.dim[3].pow(3));
186                     BigDecimal b = this.dim[0].subtract(this.dim[2].multiply(new
    BigDecimal("2")));
187                     BigDecimal c = this.dim[1].subtract(this.dim[3].multiply(new
    BigDecimal("2"))).pow(3);
188                     I = a.subtract(b.multiply(c)).divide(new BigDecimal(12), p,
    RoundingMode.HALF_EVEN);
189                 }; break;
190                 //Caso 5: Seção T.
191                 case 6 : {
192                     //Y do coentroide de cada componente
193                     BigDecimal y1 = this.dim[1].divide(new BigDecimal("2"),p);
194                     BigDecimal y2 = this.dim[3].divide(new BigDecimal("2"),p);
195                     //Áreas dos componentes
196                     BigDecimal A1 = this.dim[1].multiply(this.dim[2]);
197                     BigDecimal A2 =
    (this.dim[0].subtract(this.dim[2])).multiply(this.dim[3]);
198                     //Centroide da seção
199                     BigDecimal y =
    (A1.multiply(y1).add(A2.multiply(y2))).divide(A1.add(A2),p);
200                     //Inércia dos componentes em relação ao centroide da seção
201                     BigDecimal I1 = this.dim[2].multiply(this.dim[1].pow(3)).divide(new
    BigDecimal("12"), p, RoundingMode.HALF_EVEN).
202                                         add(y1.subtract(y).pow(2));
203                     BigDecimal I2 =
    this.dim[0].subtract(this.dim[2]).multiply(this.dim[3].pow(3)).divide(new BigDecimal("12"),
    p, RoundingMode.HALF_EVEN).
204                                         add(A2.multiply(y2.subtract(y).pow(2)));
205                     I = I2.add(I1);
206                 }; break;
207                 case 7 : {
208                     //Y do centroide de cada componente
209                     BigDecimal y2 = this.dim[3].divide(new BigDecimal("2"), p,
    RoundingMode.HALF_EVEN);
210                     //Áreas dos componentes
211                     BigDecimal A2 =
    this.dim[0].subtract(this.dim[2]).multiply(this.dim[3]);
212                     //Inércia dos componentes em relação ao centroide da seção
213                     BigDecimal I1 = this.dim[2].multiply(this.dim[1].pow(3)).divide(new
    BigDecimal("12"), p, RoundingMode.HALF_EVEN);
214                     BigDecimal I2 =
    this.dim[0].subtract(this.dim[2]).multiply(this.dim[3].pow(3)).divide(new BigDecimal(12),
    p, RoundingMode.HALF_EVEN).
215                                             add(A2.multiply(y2.subtract(this.dim[1].divide(new
    BigDecimal("2"), p, RoundingMode.HALF_EVEN))).pow(2));
216                     //Inércia da seção
217                     I = I1.add(I2.multiply(new BigDecimal("2")));
218                 }
219             }
220             return I;
221         }
```

```
222     }
223
224
```

D.7 Classe Apoio.java

Apoio.java

```
1 package com.tcc.ecalc;
2
3 public class Apoio {
4
5     public static int getApoioId(boolean rX, boolean rY, boolean rZ) {
6             int count = 0;
7             if(rX) {count += 1;}
8             if(rY) {count += 2;}
9             if(rZ) {count += 4;}
10            return count;
11    }
12
13    public static boolean[] getRestricoes(int id) {
14        boolean[] restricoes = new boolean[]{false,false,false};
15        switch(id) {
16            case 0 :
17                break;
18            case 1 :
19                restricoes[0] = true;
20                break;
21            case 2 :
22                restricoes[1] = true;
23                break;
24            case 3 :
25                restricoes[0] = true;
26                restricoes[1] = true;
27                break;
28            case 4 :
29                restricoes[2] = true;
30                break;
31            case 5 :
32                restricoes[0] = true;
33                restricoes[2] = true;
34                break;
35            case 6 :
36                restricoes[1] = true;
37                restricoes[2] = true;
38                break;
39            case 7 :
40                restricoes[0] = true;
41                restricoes[1] = true;
42                restricoes[2] = true;
43                break;
44        }
45        return restricoes;
46    }
47 }
48
49
```

D.8 Classe No.java

No.java

```
 1 package com.tcc.ecalc;
 2
 3 public class No {
 4     private int _id, //Id
 5                 Cp; //Carga Pontual
 6     private Ponto C = new Ponto(); //Coordenadas
 7     private boolean[] R = new boolean[3]; //Restrições de apoio
 8     private boolean selec,
 9                     rot; //Estado de seleção
10
11     //Construtor de Nó
12     public No(int _id, double x, double y, int Cp, boolean Rx, boolean Ry, boolean Rz,
   boolean rot) {
13         this._id = _id;
14         this.C.setX(x);
15         this.C.setY(y);
16         this.R[0] = Rx;
17         this.R[1] = Ry;
18         this.R[2] = Rz;
19         this.Cp = Cp;
20         this.selec = false;
21         this.rot = rot;
22     }
23
24     public No(No n) {
25         this(n._id,n.C.getX(),n.C.getY(),n.Cp,n.R[0],n.R[1],n.R[2],n.rot);
26     }
27
28     //Retorna uma string inteligível descrevendo o nó
29     public String toString() {
30         return("id = "+this._id+", X = "+this.getC().getX()+
31                 ", Y = "+this.getC().getY()+", Cp = "+this.getCp()+", Rx =
   "+this.getRestricao(0)+
32                 ", Ry = "+this.getRestricao(1)+", Rz = "+this.getRestricao(2)+", n =
   "+this.getApNumero());
33     }
34
35     //Retorna o número de caracterização do apoio
36     public int getApNumero() {
37         int I = 0;
38         I = ((R[0])? 1:0) +
39             ((R[1])? 2:0) +
40             ((R[2])? 4:0);
41         return I;
42     }
43
44     //-------------------------------------------------------------------------------Getter
   s
45
46     public boolean getSelection() {
47         return this.selec;
48     }
49
50     public int getId() {
51         return this._id;
52         }
53
54     public static Ponto getC(No No) {
55         return No.C;
56         }
57
```

```
58      public boolean getRot() {
59          return rot;
60      }
61
62      public Ponto getC() {
63          return this.C;
64          }
65
66      public boolean getRestricao(int i) {
67          return R[i];
68          }
69
70      public boolean[] getRestricao() {
71          return R;
72          }
73
74      public int getCp() {
75          return Cp;
76          }
77
78      //-------------------------------------------------------------------------------Setter
    s
79
80      public void setSelection(boolean selec) {
81          this.selec = selec;
82      }
83
84      public void setRot(boolean rot) {
85          this.rot = rot;
86      }
87
88      public void setRestricao(boolean Rx, boolean Ry, boolean Rz) {
89          this.R[0] = Rx;
90          this.R[1] = Ry;
91          this.R[2] = Rz;
92      }
93
94      public void setCPon(int CPon) {
95          this.Cp = CPon;
96      }
97  }
98
99
```

D.9 Classe Barra.java

Barra.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.math.BigDecimal;
 6
 7 public class Barra {
 8
 9     private int ID,         //Id da barra
10                 N1,         //Id do primeiro nó
11                 N2,         //Id do segundo nó
12                 Material,   //Id do material
13                 Perfil,     //Id do perfil
14                 Q;          //Id da carga distribuida
15     private BigDecimal[] esfInt = new BigDecimal[6]; //Reações na barra
16     private BigDecimal[] desLocal = new BigDecimal[6]; //Reações na barra
17     private double angulo, //Ângulo da barra com a horizontal
18                    L;      //Largura da barra
19     private boolean selec; //Estado de seleção da barra
20     private boolean[] rot;
21     final public static int NO_ESQUERDO = 0;
22     final public static int  NO_DIREITO = 1;
23
24     //Construtor de Barra
25     Barra(int ID, int N1,int N2 ,int Perfil,int Material,int Q,boolean rotE,boolean rotD) {
26         //Atribui os parâmetros de criação para as variáveis do objeto
27         this.ID = ID;
28         this.N1 = N1;
29         this.N2 = N2;
30         this.Material = Material;
31         this.Perfil = Perfil;
32         this.Q = Q;
33         this.selec = false;
34         this.rot = new boolean[]{rotE,rotD};
35     }
36
37     Barra(Barra b) {
38         this.ID = b.ID;
39         this.N1 = b.N1;
40         this.N2 = b.N2;
41         this.Material = b.Material;
42         this.Perfil = b.Perfil;
43         this.Q = b.Q;
44         this.selec = false;
45         this.rot = new boolean[]{b.rot[NO_ESQUERDO],b.rot[NO_DIREITO]};
46         this.angulo= b.angulo;
47         this.L =  b.L;
48     }
49
50     @Override //Retorna uma string inteligível descrevendo o elemento de barra
51     public String toString() {
52         return "id = "+ID+", N1 = "+N1+", N2 = "+N2+", Material = "+Material+", Perfil =
   "+Perfil+", Q = "+Q+
53                 ", L = "+L+", A = "+angulo+"º, rotE = "+rot[0]+", rotD = "+rot[1];
54     }
55
56     //Pega as coordenadas dos nós
57     public Ponto[] getNos(No[] Nos,Ponto[] NosB) {
58         for(int i = 0; i < Nos.length; i++) {
59             if(Nos[i].getId() == this.N1) {
60                NosB[0].setX(Nos[i].getC().getX());
61                NosB[0].setY(Nos[i].getC().getY());
62             }
```

```
63              if(Nos[i].getId() == this.N2) {
64                 NosB[1].setX(Nos[i].getC().getX());
65                 NosB[1].setY(Nos[i].getC().getY());
66              }
67          }
68          return NosB;
69      }
70
71      // ------------------------------------------------------------------------ Getters
72
73      public int getId() {
74          return this.ID;
75      }
76
77      public double getL() {
78          return this.L;
79      }
80
81      public int getQ() {
82          return this.Q;
83      }
84
85      public int getN(int NO) {
86          if(NO == NO_ESQUERDO) {
87              return this.N1;}
88          else {
89              if(NO== NO_DIREITO) {
90               return this.N2;}
91          }
92          Log.d("Exception","Erro ao pegar nó = "+NO);
93          return -1;
94      }
95
96      public double getAngulo() {
97          return this.angulo;
98      }
99
100     public boolean getSelection() {
101         return this.selec;
102     }
103
104     public boolean[] getRot() {
105         return this.rot;
106     }
107
108     public boolean getNoRot(No[] Nos, int id) {
109         for(int i = 0; i < Nos.length; i++) {
110             if(Nos[i].getId()==id) {
111                 return Nos[i].getRot();
112             }
113         }
114         return false;
115     }
116
117     public int getMat() {
118         return Material;
119     }
120
121     public int getPerfil() {
122         return Perfil;
123     }
```

```
124
125     public boolean getRot(int i) {
126         return this.rot[i];
127     }
128
129     // ------------------------------------------------------------------------ Setters
130
131     public void setL(double L) {
132         this.L = L;
133     }
134
135     public void setQ(int Q) {
136         this.Q = Q;
137     }
138
139     public void setPerfil(int perfil) {
140         this.Perfil = perfil;
141     }
142
143     public void setMaterial(int material) {
144         this.Material = material;
145     }
146
147     public void setAngulo(double A) {
148         this.angulo = A;
149     }
150
151     public void setSelection(boolean selec) {
152         this.selec = selec;
153     }
154
155     public void setRot(boolean rotE, boolean rotD) {
156         this.rot = new boolean[]{rotE, rotD};
157     }
158
159     public void setRot(int no, boolean rot) {
160         this.rot[no] = rot;
161     }
162
163     public void setEsforcosInternos(BigMatrix esfInt) {
164         for(int i = 0; i < 6; i++) {
165             this.esfInt[i] = esfInt.get(i, 0);
166         }
167     }
168
169     public BigDecimal getEsforcosInternos(int index) {
170         return this.esfInt[index];
171     }
172
173     public void setDesLocal(BigMatrix desLocal) {
174         for(int i = 0; i < 6; i++) {
175             this.desLocal[i] = desLocal.get(i, 0);
176         }
177     }
178
179     public BigDecimal getDesLocal(int i) {
180         return this.desLocal[i];
181     }
182 }
```

D.10 Classe Estrutura.java

Estrutura.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.io.OutputStreamWriter;
12
13 public class Estrutura {
14
15     public Barra[] mBarras;
16     public No[] mNos;
17     public CargaDistribuida[] mCDis;
18     public CargaPontual[] mCPon;
19     public Material[] mMateriais;
20     public Perfil[] mPerfis;
21     public BigMatrix mRigidez;
22     public BigMatrix mEngPerfeito;
23     public BigMatrix mDeslocamentos;
24     public BigMatrix mAcoesNodais;
25     public BigMatrix mReacoesNodais;
26     private DataHandler handler;
27     private Context ctx;
28     private OutputStreamWriter mWriter;
29
30     public static final int p = 30;
31
32     public Estrutura(String dataBase, Context ctx) {
33         this.ctx = ctx;
34
35         handler = new DataHandler(this.ctx);
36         handler.open();
37
38         try{
39             mWriter = new
   OutputStreamWriter(ctx.openFileOutput("calc.txt",Context.MODE_PRIVATE));
40         } catch (Exception e) {
41             Log.d("Exception", "File not Found");
42             e.printStackTrace();
43         }
44
45         populateAll();
46
47         mRigidez = new BigMatrix(0,0, p);
48         mEngPerfeito = new BigMatrix(0,0, p);
49         mDeslocamentos = new BigMatrix(0,0, p);
50         mAcoesNodais = new BigMatrix(0,0, p);
51
52     }
53
54
55     //Add
56
57     public void addDistribuida(CargaDistribuida cDis) {
58         String[] keys = new String[]{"_id","nome","Qx0","Qy0","Qx1","Qy1","isGlobal"};
59         String[] data = new String[]{String.valueOf(cDis.getID()),
60                                                 cDis.getNome(),
61                                     String.valueOf(cDis.getQ(0)),
62                                     String.valueOf(cDis.getQ(1)),
63                                     String.valueOf(cDis.getQ(2)),
64                                     String.valueOf(cDis.getQ(3)),
65                                     String.valueOf(cDis.isGlobal()? 1 : 0)};
66         handler.insertData("CARGASDISTRIBUIDAS", keys, data);
67         populateDistribuidas();
68     }
```

```
69
70      public void addPontual(CargaPontual cPon) {
71          String[] keys = new String[]{"_id","nome","Px","Py","Pz"};
72          String[] data = new String[]{String.valueOf(cPon.getId()),
73                                                  cPon.getNome(),
74                                      String.valueOf(cPon.getP(0)),
75                                      String.valueOf(cPon.getP(1)),
76                                      String.valueOf(cPon.getP(2))};
77          handler.insertData("CARGASPONTUAIS", keys, data);
78          populatePontuais();
79      }
80
81      public void addPerfil(Perfil perfil) {
82          String[] keys = new
    String[]{"_id","nome","tipo","aSec","momI","dim0","dim1","dim2","dim3"};
83          String[] data = new String[]{String.valueOf(perfil.getId()),
84                                                  perfil.getNome(),
85                                      String.valueOf(perfil.getTipo()),
86                                      String.valueOf(perfil.getASec()),
87                                      String.valueOf(perfil.getMomI()),
88                                      String.valueOf(perfil.getDim(0)),
89                                      String.valueOf(perfil.getDim(1)),
90                                      String.valueOf(perfil.getDim(2)),
91                                      String.valueOf(perfil.getDim(3))};
92          handler.insertData("PERFIS", keys, data);
93          populatePerfis();
94      }
95
96      public void addMaterial(Material material) {
97          String[] keys = new String[]{"_id","nome","modE","dVol"};
98          String[] data = new String[]{String.valueOf(material.getId()),
99                                                  material.getNome(),
100                                     String.valueOf(material.getModE()),
101                                     String.valueOf(material.getDVol())};
102         handler.insertData("MATERIAIS", keys, data);
103         populateMateriais();
104     }
105
106     public boolean addBarra(Ponto A, Ponto B) {
107         /*if(!(A.getAngulo(B) > -90 && A.getAngulo(B) <= 90)) {
108             addBarra(B, A);
109             return false;
110         }*/
111
112         boolean aExists = false, bExists = false;
113         int N1 = -1, N2 = -1;
114         for(int i = 0; i < mBarras.length; i++) {
115             if(Ponto.coinc(A, B, mNos[mBarras[i].getN(Barra.NO_ESQUERDO)].getC(),
116                                  mNos[mBarras[i].getN(Barra.NO_DIREITO)].getC())) {
117                     return false;}
118         }
119         if(A.isEqual(B)) {
120             return false;} else {
121                 for(int i = 0; i < mNos.length; i++) {
122                     if(mNos[i].getC().isEqual(A)) {
123                         aExists = true;
124                         handler.execSQL("UPDATE NOS SET rot = 0 WHERE _id = "+i);
125                         N1 = i;
126                     } else {
127                         if(mNos[i].getC().isEqual(B)) {
128                             bExists = true;
```

```
129                             handler.execSQL("UPDATE NOS SET rot = 0 WHERE _id = "+i);
130                             N2 = i;
131                         }
132                     }
133                 }
134                 if(!aExists) {
135                     String query;
136                     if(mNos.length == 0) {
137                         query = "INSERT INTO NOS(_id,x,y) VALUES("+0+","+A.getX()+",
    "+A.getY()+")";}
138                     else {
139                         query = "INSERT INTO NOS(x,y) VALUES("+A.getX()+", "+A.getY()+")";
140                     }
141                     handler.execSQL(query);
142                     N1 = mNos.length;
143                 }
144                 if(!bExists) {
145                     String query = "INSERT INTO NOS(x,y) VALUES("+B.getX()+",
    "+B.getY()+")";
146                     handler.execSQL(query);
147                     if(aExists) {
148                         N2 = mNos.length;
149                     } else {
150                         N2 = mNos.length+1;
151                     }
152                 }
153                 populateNos();
154                 for(int i = 0; i < mBarras.length; i++) {
155                     Ponto eB = mNos[mBarras[i].getN(Barra.NO_ESQUERDO)].getC();
156                     Ponto dB = mNos[mBarras[i].getN(Barra.NO_DIREITO)].getC();
157                     Ponto C = null;
158                     C = Ponto.cruzamento(A, B, eB, dB, C);
159                     if(C != null) {
160                         if(!C.isEqual(eB) && !C.isEqual(dB)) {
161                             String query = "DELETE FROM BARRAS WHERE _id =
    "+mBarras[i].getId();
162                             handler.execSQL(query);
163                             populateBarras();
164                             addBarra(eB,C);
165                             addBarra(C,dB);
166                         }
167                         if(!C.isEqual(A) && !C.isEqual(B)) {
168                             addBarra(A,C);
169                             addBarra(C,B);
170                             return true;
171                         }
172                     }
173                 }
174             }
175 
176         String query;
177         if(mBarras.length == 0) {
178             query = "INSERT INTO BARRAS(_id,no0,no1) VALUES("+0+","+N1+","+N2+")";
179         } else {
180             query = "INSERT INTO BARRAS(no0,no1) VALUES("+N1+","+N2+")";
181         }
182         handler.execSQL(query);
183         populateNos();
184         populateBarras();
185         return true;
186     }
```

```
187
188     public void excDistribuida(int id) {
189         String[] conditionsKey = new String[]{"_id"};
190         String[] conditionsValue = new String[]{String.valueOf(id)};
191         handler.deleteData("CARGASDISTRIBUIDAS", conditionsKey, conditionsValue);
192         handler.pullOneUp("CARGASDISTRIBUIDAS","_id", id);
193 
194         String[] columns = new String[]{"cDis"};
195         String[] newValue = new String[]{"0"};
196         conditionsKey = new String[]{"cDis"};
197         handler.changeData("BARRAS", columns, newValue, conditionsKey, conditionsValue);
198         handler.pullOneUp("BARRAS", "cDis", id);
199 
200         populateDistribuidas();
201         populateBarras();
202     }
203 
204     public void excPontual(int id) {
205         String[] conditionsKey = new String[]{"_id"};
206         String[] conditionsValue = new String[]{String.valueOf(id)};
207         handler.deleteData("CARGASPONTUAIS", conditionsKey, conditionsValue);
208         handler.pullOneUp("CARGASPONTUAIS","_id", id);
209 
210         String[] columns = new String[]{"cPon"};
211         String[] newValue = new String[]{"0"};
212         conditionsKey = new String[]{"cPon"};
213         handler.changeData("NOS", columns, newValue, conditionsKey, conditionsValue);
214         handler.pullOneUp("NOS", "cPon", id);
215 
216         populatePontuais();
217         populateNos();
218     }
219 
220     public void excPerfil(int id) {
221         String[] conditionsKey = new String[]{"_id"};
222         String[] conditionsValue = new String[]{String.valueOf(id)};
223         handler.deleteData("PERFIS", conditionsKey, conditionsValue);
224         handler.pullOneUp("PERFIS","_id", id);
225 
226         String[] columns = new String[]{"nPer"};
227         String[] newValue = new String[]{"0"};
228         conditionsKey = new String[]{"nPer"};
229         handler.changeData("BARRAS", columns, newValue, conditionsKey, conditionsValue);
230         handler.pullOneUp("BARRAS", "nPer", id);
231 
232         populatePerfis();
233         populateBarras();
234     }
235 
236     public void excMaterial(int id) {
237         String[] conditionsKey = new String[]{"_id"};
238         String[] conditionsValue = new String[]{String.valueOf(id)};
239         handler.deleteData("MATERIAIS", conditionsKey, conditionsValue);
240         handler.pullOneUp("MATERIAIS","_id", id);
241 
242         String[] columns = new String[]{"nMat"};
243         String[] newValue = new String[]{"0"};
244         conditionsKey = new String[]{"nMat"};
245         handler.changeData("BARRAS", columns, newValue, conditionsKey, conditionsValue);
246         handler.pullOneUp("BARRAS", "nMat", id);
247 
```

```
248         populateMateriais();
249         populateBarras();
250     }
251 
252     public void excBarra(List<Integer> iIds) {
253         StringBuilder query = new StringBuilder();
254         query.append("DELETE FROM BARRAS WHERE _id = "+iIds.get(0));
255         for(int i = 1; i < iIds.size(); i++) {
256             query.append(" OR _id = "+iIds.get(i));
257         }
258         handler.execSQL(query.toString());
259 
260         populateBarras();
261         reindex();
262         excNoSozinho();
263     }
264 
265     public void excNo(List<Integer> iIds) {
266         StringBuilder query1 = new StringBuilder();
267         StringBuilder query2 = new StringBuilder();
268         query1.append("DELETE FROM NOS WHERE _id = "+iIds.get(0));
269         query2.append("DELETE FROM BARRAS WHERE no0 = "+iIds.get(0)+" OR no1 =
    "+iIds.get(0));
270         for(int i = 1; i < iIds.size(); i++) {
271             query1.append(" OR _id = "+iIds.get(i));
272             query2.append(" OR no0 = "+iIds.get(i)+" OR no1 = "+iIds.get(i));
273         }
274         handler.execSQL(query1.toString());
275         handler.execSQL(query2.toString());
276 
277         reindex();
278         excNoSozinho();
279     }
280 
281     public void excNoSozinho() {
282         List<Integer> iIds = new ArrayList<Integer>();
283         for(int i = 0; i < mNos.length; i++) {
284             boolean hasBarra = false;
285             for(int j = 0; j < mBarras.length; j++) {
286                 if(mBarras[j].getN(Barra.NO_ESQUERDO) == i ||
    mBarras[j].getN(Barra.NO_DIREITO) == i) {
287                     hasBarra = true;
288                     break;
289                 }
290             }
291             if(!hasBarra) {
292                 iIds.add(i);
293             }
294         }
295         StringBuilder query = new StringBuilder();
296         if(iIds.size() != 0) {
297             query.append("DELETE FROM NOS WHERE _id = "+iIds.get(0));
298             for(int i = 0; i < iIds.size(); i++) {
299                 query.append(" OR _id = "+iIds.get(i));
300             }
301             handler.execSQL(query.toString());
302             reindex();
303         }
304     }
305 
306 
```

```
307     public void setDistribuida(ArrayList<Integer> selec, int id) {
308         StringBuilder query = new StringBuilder();
309         query.append("UPDATE BARRAS SET cDis = "+id+" WHERE _id = "+selec.get(0));
310         for(int i = 1; i < selec.size(); i++) {
311             query.append(" OR _id = "+selec.get(i));
312         }
313         Log.d("Ecalc",query.toString());
314         handler.execSQL(query.toString());
315 
316         populateBarras();
317     }
318 
319     public void setAllDistribuida(int id) {
320         handler.changeData("BARRAS",new String[]{"cDis"}, new
    String[]{String.valueOf(id)});
321         populateBarras();
322     }
323 
324     public void setAllPontual(int id) {
325         handler.changeData("NOS",new String[]{"cPon"}, new String[]{String.valueOf(id)});
326         Log.d("Ecalc","Changing loading to "+id);
327         populateNos();
328     }
329 
330     public void setPontual(ArrayList<Integer> selec, int id) {
331         StringBuilder query = new StringBuilder();
332         query.append("UPDATE NOS SET cPon = "+id+" WHERE _id = "+selec.get(0));
333         for(int i = 1; i < selec.size(); i++) {
334             query.append(" OR _id = "+selec.get(i));
335         }
336         Log.d("Ecalc",query.toString());
337         handler.execSQL(query.toString());
338         populateNos();
339     }
340 
341     public void setAllApoio(int id) {
342         boolean[] R = Apoio.getRestricoes(id);
343         String[] columns = new String[]{"rX","rY","rZ"};
344         String[] newValue = new String[]{R[0]? "1" : "0",
345                                          R[1]? "1" : "0",
346                                          R[2]? "1" : "0"};
347         handler.changeData("NOS", columns, newValue);
348         populateNos();
349     }
350 
351     public void setApoio(ArrayList<Integer> selec, int id) {
352         boolean[] R = Apoio.getRestricoes(id);
353         StringBuilder query = new StringBuilder();
354         query.append("UPDATE NOS SET rX = "+(R[0]? "1":"0"));
355         query.append(", rY = "+(R[1]? "1" : "0"));
356         query.append(", rZ = "+(R[2]? "1" : "0"));
357         query.append(" WHERE _id = "+selec.get(0));
358         for(int i = 1; i < selec.size(); i++) {
359             query.append(" OR _id = "+mNos[selec.get(i)].getId());
360         }
361         handler.execSQL(query.toString());
362         populateNos();
363     }
364 
365     public void setAllPerfil(int id) {
366         handler.changeData("BARRAS",new String[]{"nPer"}, new
```

```
   String[]{String.valueOf(id)});
367         populateBarras();
368     }
369
370     public void setPerfil(ArrayList<Integer> selec, int id) {
371         StringBuilder query = new StringBuilder();
372         query.append("UPDATE BARRAS SET nPer = "+id+" WHERE _id = "+selec.get(0));
373         for(int i = 1; i < selec.size(); i++) {
374             query.append(" OR _id = "+selec.get(i));
375         }
376         Log.d("Ecalc",query.toString());
377         handler.execSQL(query.toString());
378         populateBarras();
379     }
380
381     public void setAllMaterial(int id) {
382         handler.changeData("BARRAS",new String[]{"nMat"}, new
   String[]{String.valueOf(id)});
383         populateBarras();
384     }
385
386     public void setMaterial(ArrayList<Integer> selec, int id) {
387         StringBuilder query = new StringBuilder();
388         query.append("UPDATE BARRAS SET nMat = "+id+" WHERE _id = "+selec.get(0));
389         for(int i = 1; i < selec.size(); i++) {
390             query.append(" OR _id = "+selec.get(i));
391         }
392         Log.d("Ecalc",query.toString());
393         handler.execSQL(query.toString());
394         populateBarras();
395     }
396
397     public void populateDistribuidas() {
398         Cursor linha = handler.returnData("CARGASDISTRIBUIDAS");
399         mCDis = new CargaDistribuida[linha.getCount()];
400
401         while(linha.moveToNext()) {
402             mCDis[linha.getPosition()] = new CargaDistribuida(linha.getInt(0),
403                                                               linha.getString(1),
404                                                               linha.getDouble(2),
405                                                               linha.getDouble(3),
406                                                               linha.getDouble(4),
407                                                               linha.getDouble(5),
408                                                              (linha.getInt(6)==1)?
   true:false);
409         }
410         linha.close();
411     }
412
413     public void populatePontuais() {
414         Cursor linha = handler.returnData("CARGASPONTUAIS");
415         mCPon = new CargaPontual[linha.getCount()];
416
417         while(linha.moveToNext()) {
418             mCPon[linha.getPosition()] = new CargaPontual(linha.getInt(0),
419                                                           linha.getString(1),
420                                                           linha.getDouble(2),
421                                                           linha.getDouble(3),
422                                                           linha.getDouble(4));
423         }
424         linha.close();
```

```
425     }
426
427     public void populateMateriais() {
428         Cursor linha = handler.returnData("MATERIAIS");
429         mMateriais = new Material[linha.getCount()];
430
431         while(linha.moveToNext()) {
432                mMateriais[linha.getPosition()] = new Material(linha.getInt(0),
433                                                               linha.getString(1),
434                                                               linha.getDouble(2),
435                                                               linha.getDouble(3));
436         }
437         linha.close();
438     }
439
440     public void populatePerfis() {
441         Cursor linha = handler.returnData("PERFIS");
442         mPerfis = new Perfil[linha.getCount()];
443
444         while(linha.moveToNext()) {
445             mPerfis[linha.getPosition()] = new Perfil(linha.getInt(0),
446                                                       linha.getString(1),
447                                                       linha.getInt(2),
448                                                       new BigDecimal(linha.getDouble(3)),
449                                                       new BigDecimal(linha.getDouble(4)),
450                                                       new BigDecimal(linha.getDouble(5)),
451                                                       new BigDecimal(linha.getDouble(6)),
452                                                       new BigDecimal(linha.getDouble(7)),
453                                                       new BigDecimal(linha.getDouble(8))
454                                                       );
455         }
456         linha.close();
457     }
458
459     public void populateNos() {
460         Cursor linha = handler.returnData("NOS");
461         mNos = new No[linha.getCount()];
462
463         while(linha.moveToNext()) {
464             mNos[linha.getPosition()] = new No(linha.getInt(0),
465                                                linha.getDouble(1),
466                                                linha.getDouble(2),
467                                                linha.getInt(3),
468                                               (linha.getInt(4)==1)? true:false,
469                                               (linha.getInt(5)==1)? true:false,
470                                               (linha.getInt(6)==1)? true:false,
471                                               (linha.getInt(7)==1)? true:false);
472         }
473         linha.close();
474     }
475
476     public void populateBarras() {
477
478         Cursor linha = handler.returnData("BARRAS");
479         mBarras = new Barra[linha.getCount()];
480         while(linha.moveToNext()) {
481             mBarras[linha.getPosition()] =     new Barra(linha.getInt(0),
482                                                      linha.getInt(1),
483                                                      linha.getInt(2),
484                                                      linha.getInt(3),
485                                                      linha.getInt(4),
```

```
                                                    linha.getInt(5),
                                                   (linha.getInt(6) == 1)? true:false,
                                                   (linha.getInt(7) == 1)? true:false);

            int[] P = new int[]{-1,-1};
            for(int i = 0; i < mNos.length; i++) {
                if(mBarras[linha.getPosition()].getN(Barra.NO_ESQUERDO) == i) {
                    P[0] = i;
                } else {
                    if(mBarras[linha.getPosition()].getN(Barra.NO_DIREITO) == i) {
                        P[1] = i;
                    }
                }
                if(P[0] != -1 && P[1] != -1) {
                    break;
                }
            }
            mBarras[linha.getPosition()].setL(mNos[P[1]].getC().getDistancia(mNos[P[0]].get
C()));
            mBarras[linha.getPosition()].setAngulo(mNos[P[1]].getC().getAngulo(mNos[P[0]].g
etC()));
        }
        linha.close();
    }

    public void populateAll() {
        populatePontuais();
        populateDistribuidas();
        populateMateriais();
        populatePerfis();
        populateNos();
        populateBarras();
    }

    public void reindex() {
        int count = 0;
        Cursor linha = handler.returnData("BARRAS", new String[]{"_id"});
        linha.moveToPosition(-1);

        while(linha.moveToNext()) {
            if(linha.getInt(0) != count) {
                handler.execSQL("UPDATE BARRAS SET _id='"+count+"' WHERE
_id='"+linha.getInt(0)+"'");
            }
            count++;
        }

        linha = handler.returnData("NOS", new String[]{"_id"});
        linha.moveToPosition(-1);
         count = 0;

        while(linha.moveToNext()) {
            if(linha.getInt(0) != count) {
                handler.execSQL("UPDATE NOS SET _id='"+count+"' WHERE
_id='"+linha.getInt(0)+"'");
                handler.execSQL("UPDATE BARRAS SET no0 = '"+count+"' WHERE
no0='"+linha.getInt(0)+"'");
                handler.execSQL("UPDATE BARRAS SET no1 = '"+count+"' WHERE
no1='"+linha.getInt(0)+"'");
            }
            count++;
```

```
541         }
542 
543         populateNos();
544         populateBarras();
545     }
546 
547     public void calcular() throws Exception {
548 
549         //Cria as matrizes com base no número de nós
550         mWriter.write("-------------------------------------------------------------------------
    ---\n"
551                       +"                          CÁLCULO DA ESTRUTURA
    \n"
552 
    +"-------------------------------------------------------------------------\n\n");
553 
554         mWriter.write("PERFIS:\n\n");
555         for(int i = 0; i < mPerfis.length; i++) {
556             mWriter.write(mPerfis[i].toString()+"\n\n");
557         }
558 
559         mWriter.write("MATERIAIS:\n\n");
560         for(int i = 0; i < mMateriais.length; i++) {
561             mWriter.write(mMateriais[i].toString()+"\n\n");
562         }
563 
564         mWriter.write("CARGAS DISTRIBUIDAS:\n\n");
565         for(int i = 0; i < mCDis.length; i++) {
566             mWriter.write(mCDis[i].toString()+"\n\n");
567         }
568 
569         mWriter.write("CARGAS PONTUAIS:\n\n");
570         for(int i = 0; i < mCPon.length; i++) {
571             mWriter.write(mCPon[i].toString()+"\n\n");
572         }
573 
574         for(int i = 0; i < mBarras.length; i++) {
575             if(mBarras[i].getMat()==0) {
576                 throw new
    Exception(ctx.getResources().getString(R.string.materialexception));
577             } else {
578                 if(mBarras[i].getPerfil()==0) {
579                     throw new
    Exception(ctx.getResources().getString(R.string.perfilexception));
580                 }
581             }
582         }
583 
584 
585         mRigidez = new BigMatrix(3*mNos.length,3*mNos.length,p,true);
586         mEngPerfeito = new BigMatrix(3*mNos.length,1,p,true);
587         mDeslocamentos = new BigMatrix(3*mNos.length,1,p,true);
588         mAcoesNodais = new BigMatrix(3*mNos.length,1,p,true);
589         mReacoesNodais = new BigMatrix(3*mNos.length,1,p,true);
590         boolean[] mRotulacoes = new boolean[mNos.length];
591 
592         //Varre a array de barras para montar as matrizes
593         for(int i = 0; i < mBarras.length; i++) {
594 
595 
596             if(mBarras[i].getRot(Barra.NO_ESQUERDO) &&
```

```
   mNos[mBarras[i].getN(Barra.NO_ESQUERDO)].getRot()) {
597                 if(!mRotulacoes[mBarras[i].getN(Barra.NO_ESQUERDO)]) {
598                     mBarras[i].setRot(Barra.NO_ESQUERDO, false);
599                     mRotulacoes[mBarras[i].getN(Barra.NO_ESQUERDO)]=true;
600                 }
601             }
602             if(mBarras[i].getRot(Barra.NO_DIREITO) &&
   mNos[mBarras[i].getN(Barra.NO_DIREITO)].getRot()) {
603                 if(!mRotulacoes[mBarras[i].getN(Barra.NO_DIREITO)]) {
604                     mBarras[i].setRot(Barra.NO_DIREITO, false);
605                     mRotulacoes[mBarras[i].getN(Barra.NO_DIREITO)]=true;
606                 }
607             }
608 
609             Log.d("Cálculo Estrutura", "Coletando dados da Barra "+i);
610             CargaDistribuida Q = mCDis[mBarras[i].getQ()];
611 
612             BigDecimal E = new BigDecimal(mMateriais[mBarras[i].getMat()].getModE())
613                                     .multiply(BigDecimal.TEN.pow(6));
614             BigDecimal I = mPerfis[mBarras[i].getPerfil()].getMomI()
615                                     .divide(BigDecimal.TEN.pow(12), p,
   RoundingMode.HALF_EVEN);
616             BigDecimal A = mPerfis[mBarras[i].getPerfil()].getASec()
617                                     .divide(BigDecimal.TEN.pow(6), p,
   RoundingMode.HALF_EVEN);//Math.pow(1000,2);
618             BigDecimal L = new BigDecimal(mBarras[i].getL());
619             BigDecimal angulo = new BigDecimal(mBarras[i].getAngulo());
620 
621             int no1 = mBarras[i].getN(0);
622             int no2 = mBarras[i].getN(1);
623             int mt = (mBarras[i].getRot(Barra.NO_ESQUERDO)? 1 : 0) +
   (mBarras[i].getRot(Barra.NO_DIREITO)? 2: 0);
624 
625             Log.d("Cálculo Estrutura", "Criando matrizes locais");
626 
627             mWriter.write(mBarras[i].toString()+"mt = "+mt+"\n\n");
628 
629             BigMatrix rigidez = BigMatrix.rigidez(E, I, A, L, mt, p);
630             mWriter.write("Rigidez:\n"+rigidez+"\n\n");
631 
632             BigMatrix rotacao = BigMatrix.rotacao(angulo, p);
633             mWriter.write("Rotação:\n"+rotacao+"\n\n");
634 
635             BigMatrix engPerfeito = BigMatrix.engPerfeit(Q, L, mt, p);
636             mWriter.write("Engastamento Perfeito:\n"+engPerfeito+"\n\n");
637 
638             rigidez = (rotacao.transpoe()).mult(rigidez).mult(rotacao);
639             mWriter.write("Rigidez Rotacionada:\n"+rigidez+"\n\n");
640 
641             engPerfeito = (rotacao.transpoe()).mult(engPerfeito);
642             mWriter.write("Engastamento Perfeito Rotacionado:\n"+engPerfeito+"\n\n");
643 
644             for(int j = 0; j < 3; j++) {
645                 for(int k = 0; k < 3; k++) {
646                     mRigidez.somaCelula(3*no1+j, 3*no1+k, rigidez.get(j, k));
647                 }
648                 for(int k = 3; k < 6; k++) {
649                     mRigidez.somaCelula(3*no1+j, 3*(no2-1)+k, rigidez.get(j, k));
650                 }
651                 mEngPerfeito.somaCelula(3*no1+j, 0, engPerfeito.get(j, 0));
652             }
```

```
653             for(int j = 3; j < 6; j++) {
654                 for(int k = 0; k < 3; k++) {
655                     mRigidez.somaCelula(3*(no2-1)+j, 3*no1+k, rigidez.get(j, k));
656                 }
657                 for(int k = 3; k < 6; k++) {
658                     mRigidez.somaCelula(3*(no2-1)+j, 3*(no2-1)+k, rigidez.get(j, k));
659                 }
660                 mEngPerfeito.somaCelula(3*(no2-1)+j, 0, engPerfeito.get(j, 0));
661             }
662         }
663 
664         mWriter.write("Engastamento Global:\n"+mEngPerfeito+"\n\n");
665         mWriter.write("Rigidez Global:\n"+mRigidez+"\n\n");
666 
667         for(int i = 0; i < mNos.length; i++) {
668             CargaPontual pontual = mCPon[mNos[i].getCp()];
669 
670             mAcoesNodais.set(3*(i+1)-3, 0, new BigDecimal(pontual.getP(0)));
671             mAcoesNodais.set(3*(i+1)-2, 0, new BigDecimal(pontual.getP(1)));
672             mAcoesNodais.set(3*(i+1)-1, 0, new BigDecimal(pontual.getP(2)));
673 
674             mDeslocamentos.set(3*i, 0, mNos[i].getRestricao(0)? new BigDecimal("0") : new
    BigDecimal("1"));
675             mDeslocamentos.set(3*i+1, 0, mNos[i].getRestricao(1)? new BigDecimal("0") : new
    BigDecimal("1"));
676             mDeslocamentos.set(3*i+2, 0, mNos[i].getRestricao(2)? new BigDecimal("0") : new
    BigDecimal("1"));
677 
678         }
679 
680         mWriter.write("Ações nodais:\n"+mAcoesNodais+"\n\n");
681         mWriter.write("Restrições nodais:\n"+mDeslocamentos+"\n\n");
682 
683         int count = 0;
684         for(int i = 0; i < mDeslocamentos.m; i++) {
685             if(mDeslocamentos.get(i, 0).compareTo(BigDecimal.ZERO) != 0) {
686                 count++;}
687         }
688 
689         Log.d("Calc","COUNT = "+count);
690 
691         BigMatrix SistA = new BigMatrix(count,count,p,true);
692         BigMatrix SistB = new BigMatrix(count,1,p,true);
693         BigMatrix SistC = new BigMatrix(count,1,p,true);
694 
695         int count2 = 0;
696 
697         for(int i = 0; i < mDeslocamentos.m; i++) {
698             if(mDeslocamentos.get(i, 0).compareTo(BigDecimal.ZERO) != 0) {
699                 SistC.set(count2, 0, new BigDecimal(i));
700                 count2++;}
701         }
702 
703         for(int i = 0; i < count; i++) {
704             for(int j = 0; j < count; j++) {
705                 SistA.set(i, j, mRigidez.get(SistC.get(i, 0).intValue(), SistC.get(j,
    0).intValue()));
706             }
707             SistB.set(i, 0, mAcoesNodais.get(SistC.get(i, 0).intValue(),0).subtract(
708                                 mEngPerfeito.get(SistC.get(i, 0).intValue(),0)));
709         }
```

```
710
711         mWriter.write("Sistema A:\n"+SistA.toString()+"\n\n");
712         mWriter.write("Sistema B:\n"+SistB.toString()+"\n\n");
713         mWriter.write("Sistema C:\n"+SistC.toString()+"\n\n");
714         mWriter.flush();
715             SistB = BigMatrix.resolve(SistA, SistB, ctx);
716             mWriter.write("Resultados:\n"+SistB+"\n\n");
717
718         for(int i = 0; i < SistB.m; i++) {
719             mDeslocamentos.set(SistC.get(i, 0).intValue(),0, SistB.get(i, 0));
720         }
721
722         mWriter.write("Deslocamentos:\n"+mDeslocamentos.toString()+"\n\n");
723         mReacoesNodais =
    (mRigidez.mult(mDeslocamentos)).soma(mEngPerfeito).subtrai(mAcoesNodais);
724         mWriter.write("Reações Nodaos:\n"+mReacoesNodais.toString()+"\n\n");
725         mWriter.flush();
726
727         populateAll();
728
729         for(int i = 0; i < mBarras.length; i++) {
730             mBarras[i].setEsforcosInternos(getEsfBarra(i));
731             mBarras[i].setDesLocal(getDefLocal(i));
732         }
733     }
734
735     public BigMatrix getEsfBarra(int i) {
736         //OutputStreamWriter writer = new
    OutputStreamWriter(ctx.openFileOutput("esfBarra"+i+".txt", Context.MODE_PRIVATE));
737         //writer.write(mBarras[i].toString()+"\n");
738
739         int n1 = mBarras[i].getN(Barra.NO_ESQUERDO);
740         int n2 = mBarras[i].getN(Barra.NO_DIREITO);
741
742         BigDecimal E = new
    BigDecimal(mMateriais[mBarras[i].getMat()].getModE()).multiply(BigDecimal.TEN.pow(6));
743         BigDecimal I =
    mPerfis[mBarras[i].getPerfil()].getMomI().divide(BigDecimal.TEN.pow(12),p,RoundingMode.HALF_
    EVEN);
744         BigDecimal A =
    mPerfis[mBarras[i].getPerfil()].getASec().divide(BigDecimal.TEN.pow(6),p,RoundingMode.HALF_E
    VEN);
745         BigDecimal L = new BigDecimal(mBarras[i].getL());
746         int mt = (mBarras[i].getRot(Barra.NO_ESQUERDO)? 1 : 0) +
    (mBarras[i].getRot(Barra.NO_DIREITO)? 2: 0);
747
748         BigMatrix rotacao = BigMatrix.rotacao(new BigDecimal(mBarras[i].getAngulo()), p);
749         BigMatrix engPerfeito = BigMatrix.engPerfeit(mCDis[mBarras[i].getQ()], L, mt, p);
750         BigMatrix rigidez = BigMatrix.rigidez(E, I, A, L, mt, p);
751         BigMatrix deslocamento = new BigMatrix(6,1,p);
752
753         deslocamento.set(0, 0, mDeslocamentos.get(3*n1,0));
754         deslocamento.set(1, 0, mDeslocamentos.get(3*n1+1,0));
755         deslocamento.set(2, 0, mDeslocamentos.get(3*n1+2,0));
756         deslocamento.set(3, 0, mDeslocamentos.get(3*n2,0));
757         deslocamento.set(4, 0, mDeslocamentos.get(3*n2+1,0));
758         deslocamento.set(5, 0, mDeslocamentos.get(3*n2+2,0));
759
760         deslocamento = rotacao.mult(deslocamento);
761
762         BigMatrix res = engPerfeito.soma(rigidez.mult(deslocamento));
```

```
763         return res;
764     }
765 
766     public BigMatrix getDefLocal(int i) {
767         BigMatrix rotacao = BigMatrix.rotacao(new BigDecimal(mBarras[i].getAngulo()), p);
768         int n1 = mBarras[i].getN(Barra.NO_ESQUERDO);
769         int n2 = mBarras[i].getN(Barra.NO_DIREITO);
770         BigMatrix deslocamento = new BigMatrix(6,1,p);
771 
772 
773         deslocamento.set(0, 0, mDeslocamentos.get(3*n1,0));
774         deslocamento.set(1, 0, mDeslocamentos.get(3*n1+1,0));
775         deslocamento.set(2, 0, mDeslocamentos.get(3*n1+2,0));
776         deslocamento.set(3, 0, mDeslocamentos.get(3*n2,0));
777         deslocamento.set(4, 0, mDeslocamentos.get(3*n2+1,0));
778         deslocamento.set(5, 0, mDeslocamentos.get(3*n2+2,0));
779 
780         deslocamento = rotacao.mult(deslocamento);
781 
782         return deslocamento;
783     }
784 
785     public void checkNosRot() {
786         ArrayList<Integer> iFalse = new ArrayList<Integer>();
787         for(int i = 0; i < mNos.length; i++) {
788             for(int j = 0; j < mBarras.length; j++) {
789                 if(mBarras[j].getN(0) == mNos[i].getId()) {
790                     if(!mBarras[j].getRot(0)) {
791                         iFalse.add(i);
792                         break;
793                     }
794                 }
795                 if(mBarras[j].getN(1) == mNos[i].getId()) {
796                     if(!mBarras[j].getRot(1)) {
797                         iFalse.add(i);
798                         break;
799                     }
800                 }
801             }
802         }
803         handler.execSQL("UPDATE NOS SET rot = 1");
804         if(iFalse.size() != 0) {
805             StringBuilder queryFalse = new StringBuilder();
806             queryFalse.append("UPDATE NOS SET rot = "+0+" WHERE _id = "+iFalse.get(0));
807             for(int i = 1; i < iFalse.size(); i++) {
808                 queryFalse.append(" OR _id = "+iFalse.get(i));
809             }
810             handler.execSQL(queryFalse.toString());
811         }
812     }
813 }
814 
```

APÊNDICE E – Código Java das Interfaces

E.1 Leiaute Carga Pontual

Activity_PontualTab.java

```
1 package com.tcc.ecalc;
2
3 import java.text.DecimalFormat;
4
5 import android.app.Fragment;
6 import android.database.Cursor;
7 import android.database.sqlite.SQLiteDatabase;
8 import android.os.Bundle;
9 import android.support.v4.widget.SimpleCursorAdapter;
10 import android.text.Editable;
11 import android.text.TextWatcher;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.View.OnClickListener;
15 import android.view.View.OnFocusChangeListener;
16 import android.view.ViewGroup;
17 import android.widget.AdapterView;
18 import android.widget.AdapterView.OnItemSelectedListener;
19 import android.widget.EditText;
20 import android.widget.ImageButton;
21 import android.widget.Spinner;
22 import android.widget.Toast;
23
24 public class Activity_PontualTab extends Fragment {
25
26     private EditText nome_frm;
27     private EditText fx_frm;
28     private EditText fy_frm;
29     private EditText mz_frm;
30     private ImageButton addPontual;
31     private ImageButton excPontual;
32     private ImageButton atPontual;
33     private ImageButton retPontual;
34     private ImageButton setPontual;
35     private ImageButton setAllPontual;
36     private Spinner spnPontual;
37     private Cursor linha;
38     private CargaPontual CargaSelecionada;
39     private DataHandler handler;
40     private DecimalFormat df = new DecimalFormat("0.00#");
41     private ActCanvas atv;
42
43     @Override
44     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
   savedInstanceState) {
45         View view = inflater.inflate(R.layout.activity_pontualtab, null);
46
47         atv = (ActCanvas) getActivity();
48         handler = new DataHandler(atv.getBaseContext());
49         handler.open();
50         nome_frm = (EditText) view.findViewById(R.id.ptl_frm_nome);
51         fx_frm = (EditText) view.findViewById(R.id.ptl_frm_fx);
52         fy_frm = (EditText) view.findViewById(R.id.ptl_frm_fy);
53         mz_frm = (EditText) view.findViewById(R.id.ptl_frm_mz);
54         addPontual = (ImageButton) view.findViewById(R.id.ptl_btn_add);
55         excPontual = (ImageButton) view.findViewById(R.id.ptl_btn_exc);
56         atPontual = (ImageButton) view.findViewById(R.id.ptl_btn_atualizar);
57         spnPontual = (Spinner) view.findViewById(R.id.ptl_spn_pontual);
58         retPontual = (ImageButton) view.findViewById(R.id.ptl_btn_ret);
59         setPontual = (ImageButton) view.findViewById(R.id.ptl_btn_set);
60         setAllPontual = (ImageButton) view.findViewById(R.id.ptl_btn_setall);
```

```
61
62          createSpnPontual(0);
63          createAddPontual();
64          createExcPontual();
65          createAtPontual();
66          createSetPontual();
67          createSetAllPontual();
68          createReturn();
69          createForms();
70
71          return view;
72      }
73
74      public void createSetAllPontual() {
75          setAllPontual.setOnClickListener(new OnClickListener() {
76
77              @Override
78              public void onClick(View v) {
79                  atv.E.setAllPontual(CargaSelecionada.getId());
80                  atv.mCanvasView.postInvalidate();
81                  atv.changeState(ActCanvas.STATE_NORMAL);
82                  getFragmentManager().popBackStack();
83              }
84          });
85      }
86
87      public void createSetPontual() {
88          setPontual.setOnClickListener(new OnClickListener() {
89
90              @Override
91              public void onClick(View v) {
92                  atv.setPontual(CargaSelecionada.getId());
93                  getFragmentManager().popBackStack();
94              }
95          });
96      }
97
98      public void createSpnPontual(int position) {
99          linha = handler.returnData("CARGASPONTUAIS");
100         String[] from = {"nome"};
101         int[] to = {R.id.spn_txv};
102         SimpleCursorAdapter ad = new SimpleCursorAdapter(atv.getBaseContext(),
    R.layout.spn, linha, from, to, 0);
103         linha.moveToPosition(position);
104         spnPontual.setAdapter(ad);
105         spnPontual.setSelection(position);
106         CargaSelecionada = new CargaPontual(linha.getInt(0),
107                                             linha.getString(1),
108                                             linha.getDouble(2),
109                                             linha.getDouble(3),
110                                             linha.getDouble(4));
111
112         spnPontual.setOnItemSelectedListener(new OnItemSelectedListener(){
113
114             @Override
115             public void onItemSelected(AdapterView<?> parent, View view,
116                     int position, long id) {
117                 linha.moveToPosition(position);
118                 CargaSelecionada = new CargaPontual(linha.getInt(0),
119                                                     linha.getString(1),
120                                                     linha.getDouble(2),
```

```
                                                    linha.getDouble(3),
                                                    linha.getDouble(4));
                nome_frm.setText(CargaSelecionada.getNome());
                fx_frm.setText(String.valueOf(CargaSelecionada.getP(0)));
                fy_frm.setText(String.valueOf(CargaSelecionada.getP(1)));
                mz_frm.setText(String.valueOf(CargaSelecionada.getP(2)));
                if(position == 0) {
                    toggleData(false);
                } else {toggleData(true);}
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {}
    });
        }

    public void createAddPontual() {
        addPontual.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                atv.E.addPontual(new CargaPontual(spnPontual.getCount(), "Nome", 0, 0, 0));
                createSpnPontual(spnPontual.getAdapter().getCount());
                atv.updateMenuIndexes();
            }
        });
    }

    public void createExcPontual() {
        excPontual.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                atv.E.excPontual(CargaSelecionada.getId());
                if(spnPontual.getSelectedItemPosition() ==
spnPontual.getAdapter().getCount()-1) {
                    createSpnPontual(spnPontual.getSelectedItemPosition()-1);} else {
                        createSpnPontual(spnPontual.getSelectedItemPosition());
                    }
                atv.mCanvasView.postInvalidate();
                atv.updateMenuIndexes();
                }
            });
    }

    public void createAtPontual() {
        atPontual.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if(CargaSelecionada.getNome().length() > 0) {
                    String[] columns = new String[]{"nome","Px","Py","Pz"};
                    String[] newValue = new String[]{CargaSelecionada.getNome(),

String.valueOf(CargaSelecionada.getP(0)),

String.valueOf(CargaSelecionada.getP(1)),

String.valueOf(CargaSelecionada.getP(2))};
                    String[] conditionsKey = new String[]{"_id"};
                    String[] conditionsValue = new
String[]{String.valueOf(CargaSelecionada.getId())};
                    handler.changeData("CARGASPONTUAIS", columns, newValue, conditionsKey,
conditionsValue);
```

```
176                     createSpnPontual(spnPontual.getSelectedItemPosition());
177                 } else {
178                     Toast.makeText(atv.getApplicationContext(), R.string.nome_invalido,
179                                    Toast.LENGTH_SHORT).show();
180                 }
181             atv.E.populatePontuais();
182             atv.mCanvasView.postInvalidate();
183             }
184         });
185     }
186
187     public void createForms() {
188
189         nome_frm.addTextChangedListener(new TextWatcher() {
190             @Override
191             public void onTextChanged(CharSequence s, int start, int before, int count) {
192                 try{
193                     CargaSelecionada.setNome(s.toString());}
194                 catch (Exception e) {}
195             }
196             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
197             public void afterTextChanged(Editable s) {}
198         });
199
200         fx_frm.addTextChangedListener(new TextWatcher() {
201             @Override
202             public void onTextChanged(CharSequence s, int start, int before, int count) {
203                     try {
204                     CargaSelecionada.setP(0,Double.parseDouble(s.toString()));
205                     } catch (Exception e) {}
206             }
207             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
208             public void afterTextChanged(Editable s) {}
209         });
210
211         fy_frm.addTextChangedListener(new TextWatcher() {
212             @Override
213             public void onTextChanged(CharSequence s, int start, int before, int count) {
214                 try {
215                     CargaSelecionada.setP(1,Double.parseDouble(s.toString()));
216                 } catch (Exception e) {}
217             }
218             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
219             public void afterTextChanged(Editable s) {}
220         });
221
222         mz_frm.addTextChangedListener(new TextWatcher() {
223             @Override
224             public void onTextChanged(CharSequence s, int start, int before, int count) {
225                 try {
226                     CargaSelecionada.setP(2,Double.parseDouble(s.toString()));
227                 } catch (Exception e) {}
228             }
229             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
230             public void afterTextChanged(Editable s) {}
231         });
232
```

```
233         fx_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
234             @Override
235             public void onFocusChange(View v, boolean hasFocus) {
236                 if(!hasFocus) {
237                     fx_frm.setText(df.format(CargaSelecionada.getP(0)));
238                 }
239             }
240         });
241 
242         fy_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
243             @Override
244             public void onFocusChange(View v, boolean hasFocus) {
245                 if(!hasFocus) {
246                     fy_frm.setText(df.format(CargaSelecionada.getP(1)));
247                 }
248             }
249         });
250 
251         mz_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
252             @Override
253             public void onFocusChange(View v, boolean hasFocus) {
254                 if(!hasFocus) {
255                     mz_frm.setText(df.format(CargaSelecionada.getP(2)));
256                 }
257             }
258         });
259     }
260 
261     protected void createReturn() {
262         retPontual.setOnClickListener(new OnClickListener() {
263 
264             @Override
265             public void onClick(View v) {
266                 atv.changeState(0);
267                 getFragmentManager().popBackStack();
268             }
269         });
270     }
271 
272     protected void toggleData(boolean isEnabled) {
273             nome_frm.setEnabled(isEnabled);
274             fx_frm.setEnabled(isEnabled);
275             fy_frm.setEnabled(isEnabled);
276             mz_frm.setEnabled(isEnabled);
277             atPontual.setEnabled(isEnabled);
278             excPontual.setEnabled(isEnabled);
279     }
280 }
281 
```

E.2 Leiaute Carga Distribuída

Activity_DistribuidaTab.java

```
1 package com.tcc.ecalc;
2
3 import java.text.DecimalFormat;
4
5 import android.app.Fragment;
6 import android.database.Cursor;
7 import android.os.Bundle;
8 import android.support.v4.widget.SimpleCursorAdapter;
9 import android.text.Editable;
10 import android.text.TextWatcher;
11 import android.view.LayoutInflater;
12 import android.view.View;
13 import android.view.View.OnClickListener;
14 import android.view.View.OnFocusChangeListener;
15 import android.view.ViewGroup;
16 import android.widget.AdapterView;
17 import android.widget.AdapterView.OnItemSelectedListener;
18 import android.widget.EditText;
19 import android.widget.ImageButton;
20 import android.widget.RadioButton;
21 import android.widget.Spinner;
22 import android.widget.Toast;
23
24 public class Activity_DistribuidaTab extends Fragment {
25
26     private DataHandler handler;
27     private Spinner spnDistribuida;
28     private EditText nome_frm;
29     private EditText qx1_frm;
30     private EditText qx2_frm;
31     private EditText qy1_frm;
32     private EditText qy2_frm;
33     private RadioButton global_btn;
34     private RadioButton local_btn;
35     private ImageButton atDistribuida;
36     private ImageButton addDistribuida;
37     private ImageButton excDistribuida;
38     private ImageButton setDistribuida;
39     private ImageButton setAllDistribuida;
40     private Cursor linha;
41     private CargaDistribuida DistribuidaSelecionada;
42     private DecimalFormat df = new DecimalFormat("0.00#");
43     private ImageButton retDistribuida;
44     private ActCanvas atv;
45
46     @Override
47     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
   savedInstanceState) {
48         View view = inflater.inflate(R.layout.activity_distribuidatab, null);
49
50         handler = new DataHandler(getActivity().getBaseContext());
51         handler.open();
52
53         spnDistribuida = (Spinner) view.findViewById(R.id.dis_spn_distribuida);
54         nome_frm = (EditText) view.findViewById(R.id.dis_frm_nome);
55         qx1_frm = (EditText) view.findViewById(R.id.dis_frm_qx1);
56         qx2_frm = (EditText) view.findViewById(R.id.dis_frm_qx2);
57         qy1_frm = (EditText) view.findViewById(R.id.dis_frm_qy1);
58         qy2_frm = (EditText) view.findViewById(R.id.dis_frm_qy2);
59         global_btn = (RadioButton) view.findViewById(R.id.dis_btn_global);
60         global_btn.setEnabled(false);
```

```
61          local_btn = (RadioButton) view.findViewById(R.id.dis_btn_local);
62          atDistribuida = (ImageButton) view.findViewById(R.id.dis_btn_atualizar);
63          addDistribuida = (ImageButton) view.findViewById(R.id.dis_btn_add);
64          excDistribuida = (ImageButton) view.findViewById(R.id.dis_btn_exc);
65          retDistribuida = (ImageButton) view.findViewById(R.id.dis_btn_ret);
66          setDistribuida = (ImageButton) view.findViewById(R.id.dis_btn_set);
67          setAllDistribuida = (ImageButton) view.findViewById(R.id.dis_btn_setall);
68          atv = (ActCanvas) getActivity();
69
70          createSpnDistribuida(0);
71          createAddDistribuida();
72          createAtDistribuida();
73          createExcDistribuida();
74          createRetDistribuida();
75          createSetDistribuida();
76          createSetAllDistribuida();
77          createForms();
78
79          return view;
80      }
81
82      public void onDetach() {
83          super.onDetach();
84          handler.close();
85      }
86
87      private void createSetDistribuida() {
88          setDistribuida.setOnClickListener(new OnClickListener() {
89
90              @Override
91              public void onClick(View v) {
92                  atv.setDistribuida(DistribuidaSelecionada.getID());
93                  getFragmentManager().popBackStack();
94              }
95          });
96      }
97
98      private void createSetAllDistribuida() {
99          setAllDistribuida.setOnClickListener(new OnClickListener() {
100
101             @Override
102             public void onClick(View v) {
103                 atv.E.setAllDistribuida(DistribuidaSelecionada.getID());
104                 atv.mCanvasView.postInvalidate();
105                 atv.changeState(ActCanvas.STATE_NORMAL);
106                 getFragmentManager().popBackStack();
107             }
108         });
109     }
110
111     private void createRetDistribuida() {
112         retDistribuida.setOnClickListener(new OnClickListener() {
113
114             @Override
115             public void onClick(View v) {
116                 getFragmentManager().popBackStack();
117                 atv.changeState(0);
118             }
119         });
120     }
121
```

```
122     private void createSpnDistribuida(int position) {
123         linha = handler.returnData("CARGASDISTRIBUIDAS");
124         String[] from = {"nome"};
125         int[] to = {R.id.spn_txv};
126         SimpleCursorAdapter ad = new SimpleCursorAdapter(getActivity().getBaseContext(),
    R.layout.spn, linha, from, to, 0);
127         spnDistribuida.setAdapter(ad);
128         linha.moveToPosition(position);
129         spnDistribuida.setSelection(position);
130         boolean isGlobal = false;
131         if(linha.getInt(6)==1) {isGlobal = true;}
132         DistribuidaSelecionada = new CargaDistribuida(linha.getInt(0),
133                                                       linha.getString(1),
134                                                       linha.getDouble(2),
135                                                       linha.getDouble(3),
136                                                       linha.getDouble(4),
137                                                       linha.getDouble(5),
138                                                       isGlobal);
139
140         spnDistribuida.setOnItemSelectedListener(new OnItemSelectedListener() {
141
142             @Override
143             public void onItemSelected(AdapterView<?> parent, View view,
144                     int position, long id) {
145                 linha.moveToPosition(position);
146                 boolean isGlobal = false;
147                 if(linha.getInt(6)==1) {isGlobal = true;}
148                 DistribuidaSelecionada = new CargaDistribuida(linha.getInt(0),
149                                                               linha.getString(1),
150                                                               linha.getDouble(2),
151                                                               linha.getDouble(3),
152                                                               linha.getDouble(4),
153                                                               linha.getDouble(5),
154                                                               isGlobal);
155                 nome_frm.setText(DistribuidaSelecionada.getNome());
156                 qx1_frm.setText(String.valueOf(DistribuidaSelecionada.getQ(0)));
157                 qy1_frm.setText(String.valueOf(DistribuidaSelecionada.getQ(1)));
158                 qx2_frm.setText(String.valueOf(DistribuidaSelecionada.getQ(2)));
159                 qy2_frm.setText(String.valueOf(DistribuidaSelecionada.getQ(3)));
160                 global_btn.setChecked(DistribuidaSelecionada.isGlobal());
161                 local_btn.setChecked(!DistribuidaSelecionada.isGlobal());
162                 if(position == 0) {toggleData(false);} else {toggleData(true);}
163             }
164             @Override
165             public void onNothingSelected(AdapterView<?> parent) {}
166         });
167     }
168
169     private void createAddDistribuida() {
170         addDistribuida.setOnClickListener(new OnClickListener() {
171             @Override
172             public void onClick(View v) {
173                 atv.E.addDistribuida(new CargaDistribuida(spnDistribuida.getCount(),
    "Nome",0,0,0,0, false));
174                 createSpnDistribuida(spnDistribuida.getAdapter().getCount());
175                 atv.updateMenuIndexes();
176             }
177         });
178     }
179
180     private void createAtDistribuida() {
```

```
181         atDistribuida.setOnClickListener(new OnClickListener() {
182             @Override
183             public void onClick(View v) {
184                 if(nome_frm.getText().toString().length() > 0) {
185                     String[] columns = new String[]{"nome","Qx0","Qy0","Qx1","Qy1"};
186                     String[] newValue = new String[]{DistribuidaSelecionada.getNome(),
187 
    String.valueOf(DistribuidaSelecionada.getQ(0)),
188 
    String.valueOf(DistribuidaSelecionada.getQ(1)),
189 
    String.valueOf(DistribuidaSelecionada.getQ(2)),
190 
    String.valueOf(DistribuidaSelecionada.getQ(3)),
191 
    String.valueOf(DistribuidaSelecionada.isGlobal())};
192                     String[] conditionsKey = new String[]{"_id"};
193                     String[] conditionsValue = new
    String[]{String.valueOf(DistribuidaSelecionada.getID())};
194                     handler.changeData("CARGASDISTRIBUIDAS", columns, newValue,
    conditionsKey, conditionsValue);
195                     createSpnDistribuida(spnDistribuida.getSelectedItemPosition());
196                 } else {
197                     Toast.makeText(getActivity().getApplicationContext(),
    R.string.nome_invalido,
198                                 Toast.LENGTH_SHORT).show();
199                 }
200                 atv.E.populateDistribuidas();
201                 atv.mCanvasView.postInvalidate();
202             }
203         });
204     }
205 
206     private void createExcDistribuida() {
207         excDistribuida.setOnClickListener(new OnClickListener() {
208             @Override
209             public void onClick(View v) {
210                 atv.E.excDistribuida(DistribuidaSelecionada.getID());
211                 if(spnDistribuida.getSelectedItemPosition() ==
    spnDistribuida.getAdapter().getCount()-1) {
212                     createSpnDistribuida(spnDistribuida.getSelectedItemPosition()-1);} else
    {
213                     createSpnDistribuida(spnDistribuida.getSelectedItemPosition()); }
214                 atv.mCanvasView.postInvalidate();
215                 atv.updateMenuIndexes();
216             }
217         });
218     }
219 
220     private void createForms() {
221         nome_frm.addTextChangedListener(new TextWatcher() {
222             @Override
223             public void onTextChanged(CharSequence s, int start, int before, int count) {
224                 try{
225                 DistribuidaSelecionada.setNome(s.toString());
226                 } catch (Exception e) {
227                 }
228             }
229             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
230             public void afterTextChanged(Editable s) {}
```

```
231         });
232
233         qx1_frm.addTextChangedListener(new TextWatcher() {
234             @Override
235             public void onTextChanged(CharSequence s, int start, int before, int count) {
236                 try {
237                 DistribuidaSelecionada.setQ(0, Double.parseDouble(s.toString()));
238                 } catch(Exception e) {}
239             }
240             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
241             public void afterTextChanged(Editable s) {}
242         });
243
244         qy1_frm.addTextChangedListener(new TextWatcher() {
245             @Override
246             public void onTextChanged(CharSequence s, int start, int before, int count) {
247                 try {
248                 DistribuidaSelecionada.setQ(1, Double.parseDouble(s.toString()));
249                 }  catch(Exception e) {}
250             }
251             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
252             public void afterTextChanged(Editable s) {}
253         });
254
255         qx2_frm.addTextChangedListener(new TextWatcher() {
256             @Override
257             public void onTextChanged(CharSequence s, int start, int before, int count) {
258                 try {
259                 DistribuidaSelecionada.setQ(2, Double.parseDouble(s.toString()));
260                 } catch(Exception e) {}
261             }
262             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
263             public void afterTextChanged(Editable s) {}
264         });
265
266         qy2_frm.addTextChangedListener(new TextWatcher() {
267             @Override
268             public void onTextChanged(CharSequence s, int start, int before, int count) {
269                 try {
270                 DistribuidaSelecionada.setQ(3, Double.parseDouble(s.toString()));
271                 } catch(Exception e) {}
272             }
273             public void beforeTextChanged(CharSequence s, int start, int count, int after)
    {}
274             public void afterTextChanged(Editable s) {}
275         });
276
277         qx1_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
278             @Override
279             public void onFocusChange(View v, boolean hasFocus) {
280                 if(!hasFocus) {
281                     qx1_frm.setText(df.format(DistribuidaSelecionada.getQ(0)));
282                 }
283             }
284         });
285
286         qy1_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
287             @Override
```

```
288             public void onFocusChange(View v, boolean hasFocus) {
289                 if(!hasFocus) {
290                     qy1_frm.setText(df.format(DistribuidaSelecionada.getQ(1)));
291                 }
292             }
293         });
294 
295         qx2_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
296             @Override
297             public void onFocusChange(View v, boolean hasFocus) {
298                 if(!hasFocus) {
299                     qx2_frm.setText(df.format(DistribuidaSelecionada.getQ(2)));
300                 }
301             }
302         });
303 
304         qy2_frm.setOnFocusChangeListener(new OnFocusChangeListener() {
305             @Override
306             public void onFocusChange(View v, boolean hasFocus) {
307                 if(!hasFocus) {
308                     qy2_frm.setText(df.format(DistribuidaSelecionada.getQ(3)));
309                 }
310             }
311         });
312     }
313 
314     private void toggleData(boolean isEnabled) {
315         nome_frm.setEnabled(isEnabled);
316         atDistribuida.setEnabled(isEnabled);
317         excDistribuida.setEnabled(isEnabled);
318         qx1_frm.setEnabled(isEnabled);
319         qy1_frm.setEnabled(isEnabled);
320         qx2_frm.setEnabled(isEnabled);
321         qy2_frm.setEnabled(isEnabled);
322         global_btn.setEnabled(isEnabled);
323         local_btn.setEnabled(isEnabled);
324     }
325 }
```

E.3 Leiaute Material

Activity_MaterialTab.java

```
 1 package com.tcc.ecalc;
 2
 3 import android.app.Fragment;
 4 import android.database.Cursor;
 5 import android.os.Bundle;
 6 import android.support.v4.widget.SimpleCursorAdapter;
 7 import android.util.Log;
 8 import android.view.LayoutInflater;
 9 import android.view.View;
10 import android.view.View.OnClickListener;
11 import android.view.ViewGroup;
12 import android.widget.AdapterView;
13 import android.widget.AdapterView.OnItemSelectedListener;
14 import android.widget.EditText;
15 import android.widget.ImageButton;
16 import android.widget.Spinner;
17
18 public class Activity_MaterialTab extends Fragment {
19
20     private Spinner spnMaterial;
21     private EditText nome;
22     private EditText modE;
23     private EditText dVol;
24     private ImageButton atMaterial;
25     private ImageButton addMaterial;
26     private ImageButton excMaterial;
27     private ImageButton retMaterial;
28     private ImageButton setMaterial;
29     private ImageButton setAllMaterial;
30     private Material MaterialSelecionado;
31     DataHandler handler;
32     private ActCanvas atv;
33
34     @Override
35     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
   savedInstanceState) {
36         View view = inflater.inflate(R.layout.activity_materialtab, null);
37         spnMaterial = (Spinner) view.findViewById(R.id.mat_spn_material);
38         nome = (EditText) view.findViewById(R.id.mat_frm_nome);
39         modE = (EditText) view.findViewById(R.id.mat_frm_elasticidade);
40         dVol = (EditText) view.findViewById(R.id.mat_frm_densidade);
41         atMaterial = (ImageButton) view.findViewById(R.id.mat_btm_atualizar);
42         addMaterial = (ImageButton) view.findViewById(R.id.mat_btn_add);
43         excMaterial = (ImageButton) view.findViewById(R.id.mat_btn_exc);
44         retMaterial = (ImageButton) view.findViewById(R.id.mat_btn_ret);
45         setMaterial = (ImageButton) view.findViewById(R.id.mat_btn_set);
46         setAllMaterial = (ImageButton) view.findViewById(R.id.mat_btn_setall);
47         atv = (ActCanvas) getActivity();
48
49         handler = new DataHandler(getActivity().getBaseContext());
50         handler.open();
51         createSpinner(0);
52         createAddMaterial();
53         createAtMaterial();
54         createExcMaterial();
55         createRetMaterial();
56         createSetMaterial();
57         createSetAllMaterial();
58         return view;
59     }
60
```

```
61      public void onDetach() {
62          super.onDetach();
63          handler.close();
64      }
65
66      public void createSetMaterial() {
67          setMaterial.setOnClickListener(new OnClickListener() {
68
69              @Override
70              public void onClick(View v) {
71                  atv.setMaterial(MaterialSelecionado.getId());
72                  getFragmentManager().popBackStack();
73              }
74          });
75      }
76
77      public void createSetAllMaterial() {
78          setAllMaterial.setOnClickListener(new OnClickListener() {
79
80              @Override
81              public void onClick(View v) {
82                  atv.E.setAllMaterial(MaterialSelecionado.getId());
83                  Log.d("Ecalc","Material = "+ MaterialSelecionado.getId());
84                  atv.changeState(ActCanvas.STATE_NORMAL);
85                  getFragmentManager().popBackStack();
86                  for(int i = 0; i < atv.E.mBarras.length; i++) {
87                      Log.d("Ecalc","Material = "+ atv.E.mBarras[i].getMat());
88                  }
89              }
90          });
91      }
92
93      public void createRetMaterial() {
94          retMaterial.setOnClickListener(new OnClickListener() {
95
96              @Override
97              public void onClick(View v) {
98                  atv.changeState(ActCanvas.STATE_NORMAL);
99                  getFragmentManager().popBackStack();
100             }
101         });
102
103     }
104
105     public void createSpinner(int position) {
106         if(position == 0) {toggleEnabled(false);} else {toggleEnabled(true);}
107         final Cursor linha = handler.returnData("MATERIAIS");
108         String[] from = {"nome"};
109         int[] to = {R.id.spn_txv};
110         SimpleCursorAdapter adp = new SimpleCursorAdapter(getActivity().getBaseContext(),
    R.layout.spn, linha, from, to, 0);
111         spnMaterial.setAdapter(adp);
112         spnMaterial.setSelection(position);
113         spnMaterial.setOnItemSelectedListener(new OnItemSelectedListener() {
114
115             @Override
116             public void onItemSelected(AdapterView<?> parent, View view,
117                     int position, long id) {
118                 linha.moveToPosition(position);
119                 MaterialSelecionado = new
    Material(linha.getInt(0),linha.getString(1),linha.getDouble(2),linha.getDouble(3));
```

```
120                 nome.setText(MaterialSelecionado.getNome());
121                 modE.setText(String.valueOf(MaterialSelecionado.getModE()));
122                 dVol.setText(String.valueOf(MaterialSelecionado.getDVol()));
123                 if(position == 0) {toggleEnabled(false);} else
124                                   {toggleEnabled(true);}
125             }
126 
127             @Override
128             public void onNothingSelected(AdapterView<?> parent) {}
129         });
130     }
131 
132     public void createAddMaterial() {
133 
134         addMaterial.setOnClickListener(new OnClickListener() {
135             @Override
136             public void onClick(View v) {
137                 atv.E.addMaterial(new Material(spnMaterial.getAdapter().getCount(), "Nome",
    0, 0));
138                 createSpinner(spnMaterial.getAdapter().getCount());
139                 atv.updateMenuIndexes();
140             }
141         });
142     }
143 
144     public void createAtMaterial() {
145 
146         atMaterial.setOnClickListener(new OnClickListener() {
147             @Override
148             public void onClick(View v) {
149                 String[] columns = new String[]{"nome","modE","dVol"};
150                 String[] newValue = new
    String[]{nome.getText().toString(),modE.getText().toString(),dVol.getText().toString()};
151                 String[] conditionsKey = new String[]{"_id"};
152                 String[] conditionsValue = new
    String[]{String.valueOf(spnMaterial.getSelectedItemId())};
153                 handler.changeData("MATERIAIS", columns, newValue, conditionsKey,
    conditionsValue);
154                 createSpinner(spnMaterial.getSelectedItemPosition());
155                 atv.E.populateMateriais();
156             }
157         });
158     }
159 
160     public void createExcMaterial() {
161         excMaterial.setOnClickListener(new OnClickListener() {
162             @Override
163             public void onClick(View v) {
164                 atv.E.excMaterial(MaterialSelecionado.getId());
165                 if(spnMaterial.getSelectedItemPosition() ==
    spnMaterial.getAdapter().getCount()-1) {
166                     createSpinner(spnMaterial.getSelectedItemPosition()-1);} else {
167                     createSpinner(spnMaterial.getSelectedItemPosition());   }
168                 atv.updateMenuIndexes();
169             }
170         });
171 
172     }
173 
174     public void toggleEnabled(boolean is) {
175         nome.setEnabled(is);
```

```
176         modE.setEnabled(is);
177         dVol.setEnabled(is);
178         atMaterial.setEnabled(is);
179         excMaterial.setEnabled(is);
180     }
181 }
182
```

E.4 Leiaute Perfil

Activity_PerfilTab.java

```
1 package com.tcc.ecalc;
2
3 import java.math.BigDecimal;
4 import java.text.DecimalFormat;
5 import java.util.jar.Attributes.Name;
6
7 import android.app.Fragment;
8 import android.content.Context;
9 import android.database.Cursor;
10 import android.graphics.Canvas;
11 import android.graphics.Paint;
12 import android.graphics.Paint.Join;
13 import android.graphics.Path;
14 import android.graphics.Point;
15 import android.graphics.PorterDuff;
16 import android.graphics.PorterDuffXfermode;
17 import android.graphics.Rect;
18 import android.os.Bundle;
19 import android.text.Editable;
20 import android.text.TextWatcher;
21 import android.util.Log;
22 import android.view.LayoutInflater;
23 import android.view.View;
24 import android.view.View.OnClickListener;
25 import android.view.ViewGroup;
26 import android.widget.AdapterView;
27 import android.widget.AdapterView.OnItemSelectedListener;
28 import android.widget.EditText;
29 import android.widget.ImageButton;
30 import android.widget.LinearLayout;
31 import android.widget.Spinner;
32 import android.widget.TextView;
33 import android.widget.Toast;
34
35 public class Activity_PerfilTab extends Fragment {
36
37     private DataHandler handler;
38     private Spinner spnPerfil;
39     private Spinner spnTipo;
40     private EditText nome;
41     private EditText d0_frm;
42     private EditText d1_frm;
43     private EditText d2_frm;
44     private EditText d3_frm;
45     private EditText momI_frm;
46     private EditText aSec_frm;
47     private TextView d0_lbl;
48     private TextView d1_lbl;
49     private TextView d2_lbl;
50     private TextView d3_lbl;
51     private TextView momI_lbl;
52     private TextView aSec_lbl;
53     private TextView d0_uni;
54     private TextView d1_uni;
55     private TextView d2_uni;
56     private TextView d3_uni;
57     private TextView momI_uni;
58     private TextView aSec_uni;
59     private ImageButton addPerfil;
60     private ImageButton excPerfil;
61     private ImageButton atPerfil;
```

```
62      private ImageButton retPerfil;
63      private ImageButton setPerfil;
64      private ImageButton setAllPerfil;
65      private Perfil PerfilSelecionado;
66      private Paint paint;
67      private Point p;
68      private DecimalFormat df;
69      private View_Canvas canvas;
70      private LinearLayout per_img;
71      private int color;
72      private ActCanvas atv;
73
74
75      @Override
76      public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
    savedInstanceState) {
77          View view = inflater.inflate(R.layout.activity_perfiltab, null);
78
79          color = getResources().getColor(R.color.Battleship_Grey);
80
81          handler = new DataHandler(getActivity().getBaseContext());
82          handler.open();
83          atv = (ActCanvas) getActivity();
84          spnPerfil = (Spinner) view.findViewById(R.id.per_spn_perfil);
85          spnTipo = (Spinner) view.findViewById(R.id.per_spn_tipo);
86          nome = (EditText) view.findViewById(R.id.per_frm_nome);
87          d0_frm = (EditText) view.findViewById(R.id.per_frm_d0);
88          d1_frm = (EditText) view.findViewById(R.id.per_frm_d1);
89          d2_frm = (EditText) view.findViewById(R.id.per_frm_d2);
90          d3_frm = (EditText) view.findViewById(R.id.per_frm_d3);
91          momI_frm = (EditText) view.findViewById(R.id.per_frm_inercia);
92          aSec_frm = (EditText) view.findViewById(R.id.per_frm_area);
93          d0_lbl = (TextView) view.findViewById(R.id.per_lbl_d0);
94          d1_lbl = (TextView) view.findViewById(R.id.per_lbl_d1);
95          d2_lbl = (TextView) view.findViewById(R.id.per_lbl_d2);
96          d3_lbl = (TextView) view.findViewById(R.id.per_lbl_d3);
97          momI_lbl = (TextView) view.findViewById(R.id.per_lbl_inercia);
98          aSec_lbl = (TextView) view.findViewById(R.id.per_lbl_area);
99          d0_uni = (TextView) view.findViewById(R.id.per_uni_d0);
100         d1_uni = (TextView) view.findViewById(R.id.per_uni_d1);
101         d2_uni = (TextView) view.findViewById(R.id.per_uni_d2);
102         d3_uni = (TextView) view.findViewById(R.id.per_uni_d3);
103         momI_uni = (TextView) view.findViewById(R.id.per_uni_inercia);
104         aSec_uni = (TextView) view.findViewById(R.id.per_uni_area);
105         addPerfil = (ImageButton) view.findViewById(R.id.per_btn_add);
106         excPerfil = (ImageButton) view.findViewById(R.id.per_btn_exc);
107         atPerfil = (ImageButton) view.findViewById(R.id.per_btn_atualizar);
108         retPerfil = (ImageButton) view.findViewById(R.id.per_btn_ret);
109         setPerfil = (ImageButton) view.findViewById(R.id.per_btn_set);
110         setAllPerfil = (ImageButton) view.findViewById(R.id.per_btn_setall);
111         per_img = (LinearLayout) view.findViewById(R.id.per_img_perfil);
112         canvas = new View_Canvas(atv.getBaseContext());
113         per_img.addView(canvas);
114         p = new Point(50,50);
115         Log.d("Ponto",p.x+", "+p.y);
116         df = new DecimalFormat("#.0000E0");
117
118     createSpnPerfil(0);
119     createForms();
120     createAtPerfil();
121     createAddPerfil();
```

```
122     createExcPerfil();
123     createRetPerfil();
124     createSetPerfil();
125     createSetAllPerfil();
126 
127     return view;
128     }
129 
130     public void createSetPerfil() {
131         setPerfil.setOnClickListener(new OnClickListener() {
132 
133             @Override
134             public void onClick(View v) {
135                 atv.setPerfil(PerfilSelecionado.getId());
136                 getFragmentManager().popBackStack();
137             }
138         });
139     }
140 
141     public void createSetAllPerfil() {
142         setAllPerfil.setOnClickListener(new OnClickListener() {
143 
144             @Override
145             public void onClick(View v) {
146                 atv.E.setAllPerfil(PerfilSelecionado.getId());
147                 atv.mCanvasView.postInvalidate();
148                 atv.changeState(ActCanvas.STATE_NORMAL);
149                 getFragmentManager().popBackStack();
150             }
151         });
152     }
153 
154     public void createRetPerfil() {
155         retPerfil.setOnClickListener(new OnClickListener() {
156 
157             @Override
158             public void onClick(View v) {
159                 atv.changeState(0);
160                 getFragmentManager().popBackStack();
161             }
162         });
163     }
164 
165     public void createSpnPerfil(int position) {
166 
167         final Cursor linha = handler.returnData("PERFIS");
168         Log.d("",linha.toString());
169         spnPerfil.setAdapter(new
    Adapter_CursorImage(getActivity().getBaseContext(),R.layout.spn_img,linha,
    1,2,0,"per_icn_"));
170         spnPerfil.setSelection(position);
171         linha.moveToPosition(position);
172         PerfilSelecionado = new Perfil(linha.getInt(0),
173                                        linha.getString(1),
174                                        linha.getInt(2),
175                                        new BigDecimal[]{new
    BigDecimal(linha.getDouble(5)),
176                                                         new
    BigDecimal(linha.getDouble(6)),
177                                                         new
    BigDecimal(linha.getDouble(7)),
```

```
178                                                     new
    BigDecimal(linha.getDouble(8))});
179         spnPerfil.setOnItemSelectedListener(new OnItemSelectedListener() {
180
181             @Override
182             public void onItemSelected(AdapterView<?> parent, View view, int position, long
    id) {
183                 linha.moveToPosition(position);
184                 PerfilSelecionado = new Perfil(linha.getInt(0),
185                                                linha.getString(1),
186                                                linha.getInt(2),
187                                                new BigDecimal[]{new
    BigDecimal(linha.getDouble(5)),
188                                                                 new
    BigDecimal(linha.getDouble(6)),
189                                                                 new
    BigDecimal(linha.getDouble(7)),
190                                                                 new
    BigDecimal(linha.getDouble(8))});
191                 nome.setText(linha.getString(1));
192                 momI_frm.setText(String.valueOf("%.4f "+PerfilSelecionado.getMomI()));
193                 aSec_frm.setText(String.valueOf("%.4f "+PerfilSelecionado.getASec()));
194                 d0_frm.setText(String.valueOf(PerfilSelecionado.getDim(0)));;
195                 d1_frm.setText(String.valueOf(PerfilSelecionado.getDim(1)));
196                 d2_frm.setText(String.valueOf(PerfilSelecionado.getDim(2)));
197                 d3_frm.setText(String.valueOf(PerfilSelecionado.getDim(3)));
198                 layoutUpdate(PerfilSelecionado.getTipo(),position);
199                 createSpnTipo(linha.getInt(2)-1);
200                 canvas.postInvalidate();
201             }
202             @Override
203             public void onNothingSelected(AdapterView<?> parent) {}
204         });
205     }
206
207     public void createSpnTipo(int pos) {
208         final Cursor linha = handler.returnData("PERFILTIPO");
209         spnTipo.setAdapter(new
    Adapter_CursorText(getActivity().getBaseContext(),R.layout.spn_img,linha,
    1,0,0,"per_icn_"));
210         spnTipo.setSelection(pos);
211
212         spnTipo.setOnItemSelectedListener(new OnItemSelectedListener() {
213             @Override
214             public void onItemSelected(AdapterView<?> parent, View view,
215                     int position, long id) {
216                 int tipo = (int) spnTipo.getSelectedItemId();
217                 PerfilSelecionado.setTipo(tipo);
218                 layoutUpdate(tipo, spnPerfil.getSelectedItemPosition());
219                 canvas.postInvalidate();
220             }
221             @Override
222             public void onNothingSelected(AdapterView<?> parent) {
223             }
224         });
225     }
226
227     public void createForms() {
228         d0_frm.addTextChangedListener(new TextWatcher() {
229             @Override
230             public void onTextChanged(CharSequence s, int start, int before, int count) {
```

```
231                 if(d0_frm.getText().toString().length() > 0) {
232                     PerfilSelecionado.setDim(0, new
    BigDecimal(d0_frm.getText().toString()));
233                 }
234                 else{
235                     PerfilSelecionado.setDim(0, BigDecimal.ZERO);
236                 }
237                 PerfilSelecionado.setASec(PerfilSelecionado.calcArea());
238                 PerfilSelecionado.setMomI(PerfilSelecionado.calcInercia());
239                 aSec_frm.setText(String.valueOf(df.format(PerfilSelecionado.getASec())));
240                 momI_frm.setText(String.valueOf(df.format(PerfilSelecionado.getMomI())));
241                 canvas.postInvalidate();
242             }
243             public void beforeTextChanged(CharSequence s, int start, int count,int after)
    {}
244             public void afterTextChanged(Editable s) {}
245         });
246
247         d1_frm.addTextChangedListener(new TextWatcher() {
248             @Override
249             public void onTextChanged(CharSequence s, int start, int before, int count) {
250                 if(d1_frm.getText().toString().length() > 0) {
251                     PerfilSelecionado.setDim(1, new
    BigDecimal(d1_frm.getText().toString()));
252                 }
253                 else{
254                     PerfilSelecionado.setDim(1, BigDecimal.ZERO);
255                 }
256                 PerfilSelecionado.setASec(PerfilSelecionado.calcArea());
257                 PerfilSelecionado.setMomI(PerfilSelecionado.calcInercia());
258                 aSec_frm.setText(String.valueOf(df.format(PerfilSelecionado.getASec())));
259                 momI_frm.setText(String.valueOf(df.format(PerfilSelecionado.getMomI())));
260                 canvas.postInvalidate();
261             }
262             public void beforeTextChanged(CharSequence s, int start, int count,int after)
    {}
263             public void afterTextChanged(Editable s) {}
264         });
265
266         d2_frm.addTextChangedListener(new TextWatcher() {
267             @Override
268             public void onTextChanged(CharSequence s, int start, int before, int count) {
269                 if(d2_frm.getText().toString().length() > 0) {
270                     PerfilSelecionado.setDim(2, new
    BigDecimal(d2_frm.getText().toString()));
271                 }
272                 else{
273                     PerfilSelecionado.setDim(2, BigDecimal.ZERO);
274                 }
275                 PerfilSelecionado.setASec(PerfilSelecionado.calcArea());
276                 PerfilSelecionado.setMomI(PerfilSelecionado.calcInercia());
277                 aSec_frm.setText(String.valueOf(df.format(PerfilSelecionado.getASec())));
278                 momI_frm.setText(String.valueOf(df.format(PerfilSelecionado.getMomI())));
279                 canvas.postInvalidate();
280             }
281             public void beforeTextChanged(CharSequence s, int start, int count,int after)
    {}
282             public void afterTextChanged(Editable s) {}
283         });
284
285         d3_frm.addTextChangedListener(new TextWatcher() {
```

```
286             @Override
287             public void onTextChanged(CharSequence s, int start, int before, int count) {
288                 if(d3_frm.getText().toString().length() > 0) {
289                     PerfilSelecionado.setDim(3, new
    BigDecimal(d3_frm.getText().toString()));
290                 }
291                 else{
292                     PerfilSelecionado.setDim(3, BigDecimal.ZERO);
293                 }
294                 PerfilSelecionado.setASec(PerfilSelecionado.calcArea());
295                 PerfilSelecionado.setMomI(PerfilSelecionado.calcInercia());
296                 aSec_frm.setText(String.valueOf(df.format(PerfilSelecionado.getASec())));
297                 momI_frm.setText(String.valueOf(df.format(PerfilSelecionado.getMomI())));
298                 canvas.postInvalidate();
299             }
300             public void beforeTextChanged(CharSequence s, int start, int count,int after)
    {}
301             public void afterTextChanged(Editable s) {}
302         });
303     }
304
305     public void createAtPerfil() {
306         atPerfil.setOnClickListener(new OnClickListener() {
307             @Override
308             public void onClick(View v) {
309                     String[] columns = new
    String[]{"nome","tipo","aSec","momI","dim0","dim1","dim2","dim3"};
310                     String[] newValue = new String[]{PerfilSelecionado.getNome(),
311
    String.valueOf(PerfilSelecionado.getTipo()),
312
    String.valueOf(PerfilSelecionado.getASec()),
313
    String.valueOf(PerfilSelecionado.getMomI()),
314
    String.valueOf(PerfilSelecionado.getDim(0)),
315
    String.valueOf(PerfilSelecionado.getDim(1)),
316
    String.valueOf(PerfilSelecionado.getDim(2)),
317
    String.valueOf(PerfilSelecionado.getDim(3))};
318                     String[] conditionsKey = new String[]{"_id"};
319                     String[] conditionsValue = new
    String[]{String.valueOf(PerfilSelecionado.getId())};
320                     handler.changeData("PERFIS", columns, newValue, conditionsKey,
    conditionsValue);
321                     createSpnPerfil(spnPerfil.getSelectedItemPosition());
322                     atv.E.populatePerfis();
323                 }
324             }
325         );
326     }
327
328     public void createAddPerfil() {
329         addPerfil.setOnClickListener(new OnClickListener() {
330             @Override
331             public void onClick(View v) {
332                 atv.E.addPerfil(new Perfil(spnPerfil.getAdapter().getCount(),"Nome", 1,
    BigDecimal.ZERO, BigDecimal.ZERO, BigDecimal.ZERO, BigDecimal.ZERO, BigDecimal.ZERO,
    BigDecimal.ZERO));
```

```
333                 createSpnPerfil(spnPerfil.getAdapter().getCount());
334                 canvas.postInvalidate();
335                 atv.updateMenuIndexes();
336             }
337         });
338     }
339 
340     public void createExcPerfil() {
341         excPerfil.setOnClickListener(new OnClickListener() {
342             @Override
343             public void onClick(View v) {
344                 atv.E.excPerfil(PerfilSelecionado.getId());
345                 if(spnPerfil.getSelectedItemPosition() ==
    spnPerfil.getAdapter().getCount()-1) {
346                     createSpnPerfil(spnPerfil.getSelectedItemPosition()-1);} else {
347                     createSpnPerfil(spnPerfil.getSelectedItemPosition());   }
348                 canvas.postInvalidate();
349                 atv.updateMenuIndexes();
350             }
351 
352         }
353         );
354     }
355 
356     public void layoutUpdate(int tipo, int position) {
357         //Atualiza o nome das labels e esconde aquelas que não são necessárias
358         if(position == 0) {
359             excPerfil.setVisibility(View.INVISIBLE);
360             atPerfil.setVisibility(View.INVISIBLE);
361             d0_lbl.setVisibility(View.INVISIBLE);
362             d1_lbl.setVisibility(View.INVISIBLE);
363             d2_lbl.setVisibility(View.INVISIBLE);
364             d3_lbl.setVisibility(View.INVISIBLE);
365             d0_frm.setVisibility(View.INVISIBLE);
366             d1_frm.setVisibility(View.INVISIBLE);
367             d2_frm.setVisibility(View.INVISIBLE);
368             d3_frm.setVisibility(View.INVISIBLE);
369             d0_uni.setVisibility(View.INVISIBLE);
370             d1_uni.setVisibility(View.INVISIBLE);
371             d2_uni.setVisibility(View.INVISIBLE);
372             d3_uni.setVisibility(View.INVISIBLE);
373             aSec_lbl.setVisibility(View.INVISIBLE);
374             aSec_frm.setVisibility(View.INVISIBLE);
375             aSec_uni.setVisibility(View.INVISIBLE);
376             momI_lbl.setVisibility(View.INVISIBLE);
377             momI_frm.setVisibility(View.INVISIBLE);
378             momI_uni.setVisibility(View.INVISIBLE);
379             spnTipo.setEnabled(false);
380             nome.setEnabled(false);
381         } else {
382             excPerfil.setVisibility(View.VISIBLE);
383             atPerfil.setVisibility(View.VISIBLE);
384         switch(tipo) {
385         //Seção geral
386         case 1: {
387             nome.setEnabled(true);
388             spnTipo.setEnabled(true);
389             atPerfil.setVisibility(View.VISIBLE);
390             //Seta os labels de acordo com a seção
391             d0_lbl.setText(R.string.area_label);
392             d1_lbl.setText(R.string.inercia_label);
```

```
393             d2_frm.setText("0.0");
394             d3_frm.setText("0.0");
395             d0_uni.setText(R.string.mm2);
396             d1_uni.setText(R.string.mm4);
397             //Altera a visibilidade dos elementos que não serão usados;
398             d0_lbl.setVisibility(View.VISIBLE);
399             d1_lbl.setVisibility(View.VISIBLE);
400             d2_lbl.setVisibility(View.INVISIBLE);
401             d3_lbl.setVisibility(View.INVISIBLE);
402             d0_frm.setVisibility(View.VISIBLE);
403             d1_frm.setVisibility(View.VISIBLE);
404             d2_frm.setVisibility(View.INVISIBLE);
405             d3_frm.setVisibility(View.INVISIBLE);
406             d0_uni.setVisibility(View.VISIBLE);
407             d1_uni.setVisibility(View.VISIBLE);
408             d2_uni.setVisibility(View.INVISIBLE);
409             d3_uni.setVisibility(View.INVISIBLE);
410             aSec_lbl.setVisibility(View.INVISIBLE);
411             aSec_frm.setVisibility(View.INVISIBLE);
412             aSec_uni.setVisibility(View.INVISIBLE);
413             momI_lbl.setVisibility(View.INVISIBLE);
414             momI_frm.setVisibility(View.INVISIBLE);
415             momI_uni.setVisibility(View.INVISIBLE);
416         } break;
417         //Seção circular
418         case 2: {
419             nome.setEnabled(true);
420             spnTipo.setEnabled(true);
421             atPerfil.setVisibility(View.VISIBLE);
422             //Seta os labels de acordo com a seção
423             d0_lbl.setText(R.string.diametro_label);
424             d1_frm.setText("0.0");
425             d2_frm.setText("0.0");
426             d3_frm.setText("0.0");
427             d0_uni.setText(R.string.mm);
428             //Altera a visibilidade dos elementos que não serão usados;
429             d0_lbl.setVisibility(View.VISIBLE);
430             d1_lbl.setVisibility(View.INVISIBLE);
431             d2_lbl.setVisibility(View.INVISIBLE);
432             d3_lbl.setVisibility(View.INVISIBLE);
433             d0_frm.setVisibility(View.VISIBLE);
434             d1_frm.setVisibility(View.INVISIBLE);
435             d2_frm.setVisibility(View.INVISIBLE);
436             d3_frm.setVisibility(View.INVISIBLE);
437             d0_uni.setVisibility(View.VISIBLE);
438             d1_uni.setVisibility(View.INVISIBLE);
439             d2_uni.setVisibility(View.INVISIBLE);
440             d3_uni.setVisibility(View.INVISIBLE);
441             aSec_lbl.setVisibility(View.VISIBLE);
442             aSec_frm.setVisibility(View.VISIBLE);
443             aSec_uni.setVisibility(View.VISIBLE);
444             momI_lbl.setVisibility(View.VISIBLE);
445             momI_frm.setVisibility(View.VISIBLE);
446             momI_uni.setVisibility(View.VISIBLE);
447         } break;
448         //Seção retangular
449         case 3: {
450             nome.setEnabled(true);
451             spnTipo.setEnabled(true);
452             atPerfil.setVisibility(View.VISIBLE);
453             //Seta os labels de acordo com a seção
```

```
454             d0_lbl.setText(R.string.base_label);
455             d1_lbl.setText(R.string.altura_label);
456             d2_frm.setText("0.0");
457             d3_frm.setText("0.0");
458             d0_uni.setText(R.string.mm);
459             d1_uni.setText(R.string.mm);
460             //Altera a visibilidade dos elementos que não serão usados;
461             d0_lbl.setVisibility(View.VISIBLE);
462             d1_lbl.setVisibility(View.VISIBLE);
463             d2_lbl.setVisibility(View.INVISIBLE);
464             d3_lbl.setVisibility(View.INVISIBLE);
465             d0_frm.setVisibility(View.VISIBLE);
466             d1_frm.setVisibility(View.VISIBLE);
467             d2_frm.setVisibility(View.INVISIBLE);
468             d3_frm.setVisibility(View.INVISIBLE);
469             d0_uni.setVisibility(View.VISIBLE);
470             d1_uni.setVisibility(View.VISIBLE);
471             d2_uni.setVisibility(View.INVISIBLE);
472             d3_uni.setVisibility(View.INVISIBLE);
473             aSec_lbl.setVisibility(View.VISIBLE);
474             aSec_frm.setVisibility(View.VISIBLE);
475             aSec_uni.setVisibility(View.VISIBLE);
476             momI_lbl.setVisibility(View.VISIBLE);
477             momI_frm.setVisibility(View.VISIBLE);
478             momI_uni.setVisibility(View.VISIBLE);
479         } break;
480         //Seção circular vazada
481         case 4: {
482             nome.setEnabled(true);
483             spnTipo.setEnabled(true);
484             atPerfil.setVisibility(View.VISIBLE);
485             //Seta os labels de acordo com a seção
486             d0_lbl.setText(R.string.diametro_externo_label);
487             d1_lbl.setText(R.string.espessura_label);
488             d2_frm.setText("0.0");
489             d3_frm.setText("0.0");
490             d0_uni.setText(R.string.mm);
491             d1_uni.setText(R.string.mm);
492             //Altera a visibilidade dos elementos que não serão usados;
493             d0_lbl.setVisibility(View.VISIBLE);
494             d1_lbl.setVisibility(View.VISIBLE);
495             d2_lbl.setVisibility(View.INVISIBLE);
496             d3_lbl.setVisibility(View.INVISIBLE);
497             d0_frm.setVisibility(View.VISIBLE);
498             d1_frm.setVisibility(View.VISIBLE);
499             d2_frm.setVisibility(View.INVISIBLE);
500             d3_frm.setVisibility(View.INVISIBLE);
501             d0_uni.setVisibility(View.VISIBLE);
502             d1_uni.setVisibility(View.VISIBLE);
503             d2_uni.setVisibility(View.INVISIBLE);
504             d3_uni.setVisibility(View.INVISIBLE);
505             aSec_lbl.setVisibility(View.VISIBLE);
506             aSec_frm.setVisibility(View.VISIBLE);
507             aSec_uni.setVisibility(View.VISIBLE);
508             momI_lbl.setVisibility(View.VISIBLE);
509             momI_frm.setVisibility(View.VISIBLE);
510             momI_uni.setVisibility(View.VISIBLE);
511         } break;
512         //Resto das seções
513         default : {
514             nome.setEnabled(true);
```

```
515             spnTipo.setEnabled(true);
516             atPerfil.setVisibility(View.VISIBLE);
517             //Seta os labels de acordo com a seção
518             d0_lbl.setText(R.string.base_label);
519             d1_lbl.setText(R.string.altura_label);
520             d2_lbl.setText(R.string.espessura_alma_label);
521             d3_lbl.setText(R.string.espessura_mesa_label);
522             d0_uni.setText(R.string.mm);
523             d1_uni.setText(R.string.mm);
524             d2_uni.setText(R.string.mm);
525             d3_uni.setText(R.string.mm);
526             //Altera a visibilidade dos elementos que não serão usados;
527             d0_lbl.setText(R.string.base_label);
528             d1_lbl.setText(R.string.altura_label);
529             d0_lbl.setVisibility(View.VISIBLE);
530             d1_lbl.setVisibility(View.VISIBLE);
531             d2_lbl.setVisibility(View.VISIBLE);
532             d3_lbl.setVisibility(View.VISIBLE);
533             d0_frm.setVisibility(View.VISIBLE);
534             d1_frm.setVisibility(View.VISIBLE);
535             d2_frm.setVisibility(View.VISIBLE);
536             d3_frm.setVisibility(View.VISIBLE);
537             d0_uni.setVisibility(View.VISIBLE);
538             d1_uni.setVisibility(View.VISIBLE);
539             d2_uni.setVisibility(View.VISIBLE);
540             d3_uni.setVisibility(View.VISIBLE);
541             aSec_lbl.setVisibility(View.VISIBLE);
542             aSec_frm.setVisibility(View.VISIBLE);
543             aSec_uni.setVisibility(View.VISIBLE);
544             momI_lbl.setVisibility(View.VISIBLE);
545             momI_frm.setVisibility(View.VISIBLE);
546             momI_uni.setVisibility(View.VISIBLE);
547         } break;
548         }
549         }
550     }
551
552     public class View_Canvas extends View {
553
554         public View_Canvas(Context context) {
555             super(context);
556             paint = new Paint();
557         }
558
559         @Override
560          protected void onSizeChanged(int xNew, int yNew, int xOld, int yOld){
561              super.onSizeChanged(xNew, yNew, xOld, yOld);
562              p = new Point(xNew, yNew);
563              Log.d("Point", p.x+", "+p.y);
564         }
565
566         public void onDraw(Canvas canvas) {
567             paint.setColor(getResources().getColor(R.color.perfil_fill));
568             paint.setStrokeWidth(5);
569             paint.setStrokeJoin(Join.ROUND);
570             Path path = new Path();
571             switch(PerfilSelecionado.getTipo()) {
572             case 1 : {} break;
573             case 2 : {
574                 float d = PerfilSelecionado.getDim(0).floatValue();
575                 canvas.drawCircle(p.x/2,p.y/2,Math.min(p.x/2,p.y/2), paint);
```

```
576             } break;
577             case 3 : {
578                 float b = PerfilSelecionado.getDim(0).floatValue();
579                 float h = PerfilSelecionado.getDim(1).floatValue();
580                 if(p.x/b < p.y/h) {
581                     float esc = p.x/b;
582                     b *= esc;
583                     h *= esc;
584                 } else {
585                     float esc = p.y/h;
586                     b *= esc;
587                     h *= esc;
588                 }
589                 path.moveTo(0, 0);
590                 path.lineTo(b, 0);
591                 path.lineTo(b, h);
592                 path.lineTo(0, h);
593                 path.close();
594                 path.offset(-(b-p.x)/2, -(h-p.y)/2);
595                 canvas.drawPath(path, paint);
596             } break;
597             case 4 : {
598                 float d = PerfilSelecionado.getDim(0).floatValue();;
599                 float t = PerfilSelecionado.getDim(1).floatValue();;
600                 float esc = Math.min(p.x, p.y)/d;
601                 d *= esc;
602                 t *= esc;
603                 canvas.drawCircle(p.x/2,p.y/2,d/2, paint);
604                 paint.setColor(color);
605                 canvas.drawCircle(p.x/2,p.y/2,(d-t)/2, paint);
606             } break;
607             case 5 : {
608                 float b = PerfilSelecionado.getDim(0).floatValue();;
609                 float h = PerfilSelecionado.getDim(1).floatValue();;
610                 float tw = PerfilSelecionado.getDim(2).floatValue();;
611                 float tf = PerfilSelecionado.getDim(3).floatValue();;
612                 if(p.x/b < p.y/h) {
613                     float esc = p.x/b;
614                     b *= esc;
615                     h *= esc;
616                     tw *= esc;
617                     tf *= esc;
618                 } else {
619                     float esc = p.y/h;
620                     b *= esc;
621                     h *= esc;
622                     tw *= esc;
623                     tf *= esc;
624                 }
625                 Rect r = new Rect(0,0,(int)b,(int)h);
626                 r.offset((int)-(b-p.x)/2,(int) -(h-p.y)/2);
627                 canvas.drawRect(r, paint);
628                 paint.setColor(color);
629                 r = new Rect((int)tw,(int)tf,(int) (b-tw),(int) (h-tf));
630                 r.offset((int)-(b-p.x)/2,(int) -(h-p.y)/2);
631                 canvas.drawRect(r, paint);
632             } break;
633             case 6 : {
634                 float b = PerfilSelecionado.getDim(0).floatValue();;
635                 float h = PerfilSelecionado.getDim(1).floatValue();;
636                 float tw = PerfilSelecionado.getDim(2).floatValue();;
```

```
637                 float tf = PerfilSelecionado.getDim(3).floatValue();;
638                 if(p.x/b < p.y/h) {
639                     float esc = p.x/b;
640                     b *= esc;
641                     h *= esc;
642                     tw *= esc;
643                     tf *= esc;
644                 } else {
645                     float esc = p.y/h;
646                     b *= esc;
647                     h *= esc;
648                     tw *= esc;
649                     tf *= esc;
650                 }
651                 path.moveTo(0,0);
652                 path.lineTo(b,0);
653                 path.lineTo(b,tf);
654                 path.lineTo((float) 0.5*(b+tw),tf);
655                 path.lineTo((float) 0.5*(b+tw),h);
656                 path.lineTo((float) 0.5*(b-tw),h);
657                 path.lineTo((float) 0.5*(b-tw),tf);
658                 path.lineTo(0,tf);
659                 path.close();
660                 path.offset(-(b-p.x)/2, -(h-p.y)/2);
661                 canvas.drawPath(path, paint);
662                 } break;
663             case 7 : {
664                 float b = PerfilSelecionado.getDim(0).floatValue();;
665                 float h = PerfilSelecionado.getDim(1).floatValue();;
666                 float tw = PerfilSelecionado.getDim(2).floatValue();;
667                 float tf = PerfilSelecionado.getDim(3).floatValue();;
668                 if(p.x/b < p.y/h) {
669                     float esc = p.x/b;
670                     b *= esc;
671                     h *= esc;
672                     tw *= esc;
673                     tf *= esc;
674                 } else {
675                     float esc = p.y/h;
676                     b *= esc;
677                     h *= esc;
678                     tw *= esc;
679                     tf *= esc;
680                 }
681                 path.moveTo(0, 0);
682                 path.lineTo(b, 0);
683                 path.lineTo(b, tf);
684                 path.lineTo((float) 0.5*(b+tw), tf);
685                 path.lineTo((float) 0.5*(b+tw), h-tf);
686                 path.lineTo(b, h-tf);
687                 path.lineTo(b, h);
688                 path.lineTo(0, h);
689                 path.lineTo(0, h-tf);
690                 path.lineTo((float) 0.5*(b-tw), h-tf);
691                 path.lineTo((float) 0.5*(b-tw), tf);
692                 path.lineTo(0, tf);
693                 path.close();
694                 path.offset(-(b-p.x)/2, -(h-p.y)/2);
695                 canvas.drawPath(path, paint);
696                 } break;
697             }
```

```
698         }
699     }
700 }
701
702
```

E.5 Leiaute Apoio

Activity_ApoioTab.java

```java
package com.tcc.ecalc;

import android.app.Fragment;

public class Activity_ApoioTab extends Fragment {

    private CheckBox Rx_btn;
    private CheckBox Ry_btn;
    private CheckBox Rz_btn;
    private ImageView apo_img;
    private ImageButton retApoio;
    private ImageButton setApoio;
    private ImageButton setAllApoio;
    private ActCanvas atv;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.activity_apoiotab, null);
        Rx_btn = (CheckBox) view.findViewById(R.id.apo_btn_Rx);
        Ry_btn = (CheckBox) view.findViewById(R.id.apo_btn_Ry);
        Rz_btn = (CheckBox) view.findViewById(R.id.apo_btn_Rz);
        apo_img = (ImageView) view.findViewById(R.id.apo_img_apoio);
        retApoio = (ImageButton) view.findViewById(R.id.apo_btn_ret);
        setApoio = (ImageButton) view.findViewById(R.id.apo_btn_set);
        setAllApoio = (ImageButton) view.findViewById(R.id.apo_btn_setall);
        atv = (ActCanvas) getActivity();

        createCheckBox();
        createRetBtn();
        createSetBtn();
        createSetAllBtn();
        imageUpdate();

        return view;
    }

    private void createRetBtn() {
        retApoio.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                getFragmentManager().popBackStack();
                atv.changeState(ActCanvas.STATE_NORMAL);
            }
        });
    }

    private void createSetAllBtn() {
        setAllApoio.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                atv.E.setAllApoio(Apoio.getApoioId(Rx_btn.isChecked(),
                                                   Ry_btn.isChecked(),
                                                   Rz_btn.isChecked()));
                atv.mCanvasView.postInvalidate();
                atv.changeState(ActCanvas.STATE_NORMAL);
                getFragmentManager().popBackStack();
            }
        });
```

```
71      }
72
73      private void createCheckBox() {
74          Rx_btn.setOnCheckedChangeListener(new OnCheckedChangeListener() {
75              @Override
76              public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
77                  imageUpdate();
78              }
79          });
80
81          Ry_btn.setOnCheckedChangeListener(new OnCheckedChangeListener() {
82              @Override
83              public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
84                  imageUpdate();
85              }
86          });
87
88          Rz_btn.setOnCheckedChangeListener(new OnCheckedChangeListener() {
89              @Override
90              public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
91                  imageUpdate();
92              }
93          });
94      }
95
96      private void createSetBtn() {
97          setApoio.setOnClickListener(new OnClickListener() {
98
99              @Override
100             public void onClick(View v) {
101                 ((ActCanvas) getActivity()).setApoio(Apoio.getApoioId(Rx_btn.isChecked(),
102                                                                       Ry_btn.isChecked(),
103                                                                       Rz_btn.isChecked()));
104                 getFragmentManager().popBackStack();
105             }
106         });
107     }
108
109     private void imageUpdate() {
110         int count =
    Apoio.getApoioId(Rx_btn.isChecked(),Ry_btn.isChecked(),Rz_btn.isChecked());
111         switch(count) {
112             case 1 : {
113                 apo_img.setBackgroundResource(R.drawable.ap_010);
114                 apo_img.setRotation(90);
115             } break;
116             case 2 : {
117                 apo_img.setBackgroundResource(R.drawable.ap_010);
118                 apo_img.setRotation(0);
119             } break;
120             case 3 : {
121                 apo_img.setBackgroundResource(R.drawable.ap_110);
122                 apo_img.setRotation(0);
123             } break;
124             case 4 : {
125                 apo_img.setBackgroundResource(R.drawable.ap_001);
126                 apo_img.setRotation(0);
127             } break;
128             case 5 : {
129                 apo_img.setBackgroundResource(R.drawable.ap_011);
130                 apo_img.setRotation(90);
```

```
131             } break;
132             case 6 : {
133                 apo_img.setBackgroundResource(R.drawable.ap_011);
134                 apo_img.setRotation(0);
135             } break;
136             case 7 : {
137                 apo_img.setBackgroundResource(R.drawable.ap_111);
138                 apo_img.setRotation(0);
139             } break;
140             default : {
141                 apo_img.setBackgroundResource(R.drawable.btn_excluielemento);
142             }
143         }
144     }
145 }
```

E.6 Leiaute Nó

Activity_NosTab.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.util.ArrayList;
 4
 5 import android.app.Fragment;
 6 import android.database.Cursor;
 7 import android.os.Bundle;
 8 import android.support.v4.widget.SimpleCursorAdapter;
 9 import android.util.Log;
10 import android.view.LayoutInflater;
11 import android.view.View;
12 import android.view.View.OnClickListener;
13 import android.view.ViewGroup;
14 import android.widget.AdapterView;
15 import android.widget.AdapterView.OnItemSelectedListener;
16 import android.widget.CheckBox;
17 import android.widget.CompoundButton;
18 import android.widget.CompoundButton.OnCheckedChangeListener;
19 import android.widget.ImageButton;
20 import android.widget.Spinner;
21
22 public class Activity_NosTab extends Fragment {
23
24     private Spinner spnPontual;
25     private CheckBox chbRx;
26     private CheckBox chbRy;
27     private CheckBox chbRz;
28     private CheckBox chbRot;
29     private ImageButton retButton;
30     private DataHandler handler;
31     private Cursor lPontual;
32     private ArrayList<Integer> nSelec;
33     private ActCanvas atv;
34
35     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
   savedInstanceState) {
36         View view = inflater.inflate(R.layout.activity_notab, null);
37
38         spnPontual = (Spinner) view.findViewById(R.id.nos_spn_ptl);
39         chbRx = (CheckBox) view.findViewById(R.id.nos_chb_rX);
40         chbRy = (CheckBox) view.findViewById(R.id.nos_chb_rY);
41         chbRz = (CheckBox) view.findViewById(R.id.nos_chb_rZ);
42         chbRot = (CheckBox) view.findViewById(R.id.nos_chb_rot);
43         retButton = (ImageButton) view.findViewById(R.id.nos_btn_ret);
44         atv = (ActCanvas) getActivity();
45         handler = new DataHandler(atv.getBaseContext());
46         nSelec = atv.getNSelec();
47
48         handler.open();
49
50         lPontual = handler.returnData("CARGASPONTUAIS");
51         String[] from = {"nome"};
52         int[] to = {R.id.spn_txv};
53
54         spnPontual.setAdapter(new
   SimpleCursorAdapter(atv.getBaseContext(),R.layout.spn,lPontual,from,to,0));
55
56         createForms();
57
58         return view;
59     }
```

```
60
61    private void createForms() {
62        spnPontual.setOnItemSelectedListener(new OnItemSelectedListener() {
63            @Override
64            public void onItemSelected(AdapterView<?> parent, View view,
65                    int position, long id) {
66                lPontual.moveToPosition(position);
67                StringBuilder query = new StringBuilder();
68                query.append("UPDATE NOS SET cPon = "+position+" WHERE _id =
  "+nSelec.get(0));
69                for(int i = 1; i < nSelec.size(); i++) {
70                        query.append(" OR _id = "+nSelec.get(i));
71                }
72                handler.execSQL(query.toString());
73                atv.E.populateNos();
74                atv.mCanvasView.postInvalidate();
75            }
76            @Override
77            public void onNothingSelected(AdapterView<?> parent) {}
78        });
79
80        chbRx.setOnCheckedChangeListener(new OnCheckedChangeListener() {
81
82            @Override
83            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
84                StringBuilder query = new StringBuilder();
85                query.append("UPDATE NOS SET rX = "+(isChecked? "1" : "0") + " WHERE _id =
  " + nSelec.get(0));
86                for(int i = 0; i < nSelec.size(); i++) {
87                    query.append(" OR _id = "+nSelec.get(i));
88                }
89                handler.execSQL(query.toString());
90                atv.E.populateNos();
91                atv.mCanvasView.postInvalidate();
92            }
93        });
94
95        chbRy.setOnCheckedChangeListener(new OnCheckedChangeListener() {
96
97            @Override
98            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
99                StringBuilder query = new StringBuilder();
100               query.append("UPDATE NOS SET rY = "+(isChecked? "1" : "0") + " WHERE _id =
  " + nSelec.get(0));
101               for(int i = 0; i < nSelec.size(); i++) {
102                   query.append(" OR _id = "+nSelec.get(i));
103               }
104               handler.execSQL(query.toString());
105               atv.E.populateNos();
106               atv.mCanvasView.postInvalidate();
107           }
108       });
109
110       chbRz.setOnCheckedChangeListener(new OnCheckedChangeListener() {
111
112           @Override
113           public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
114               StringBuilder query = new StringBuilder();
115               query.append("UPDATE NOS SET rZ = "+(isChecked? "1" : "0") + " WHERE _id =
  " + nSelec.get(0));
116               for(int i = 0; i < nSelec.size(); i++) {
```

```
117                     query.append(" OR _id = "+nSelec.get(i));
118                 }
119                 handler.execSQL(query.toString());
120                 atv.E.populateNos();
121                 atv.mCanvasView.postInvalidate();
122             }
123         });
124 
125         chbRot.setOnCheckedChangeListener(new OnCheckedChangeListener() {
126 
127             @Override
128             public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
129                 StringBuilder query1 = new StringBuilder();
130                 StringBuilder query2 = new StringBuilder();
131                 StringBuilder query3 = new StringBuilder();
132                 query1.append("UPDATE NOS SET rot = "+(isChecked? "1":"0")+" WHERE _id =
    "+nSelec.get(0));
133                 query2.append("UPDATE BARRAS SET rotE = "+(isChecked? "1":"0")+" WHERE no0
    = "+nSelec.get(0));
134                 query3.append("UPDATE BARRAS SET rotD = "+(isChecked? "1":"0")+" WHERE no1
    = "+nSelec.get(0));
135                 for(int i = 0; i < nSelec.size(); i++) {
136                     query1.append(" OR _id = "+nSelec.get(i));
137                     query2.append(" OR no0 = "+nSelec.get(i));
138                     query3.append(" OR no1 = "+nSelec.get(i));
139                 }
140                 handler.execSQL(query1.toString());
141                 handler.execSQL(query2.toString());
142                 handler.execSQL(query3.toString());
143 
144                 atv.E.populateNos();
145                 atv.E.populateBarras();
146                 atv.mCanvasView.postInvalidate();
147             }
148         });
149 
150         retButton.setOnClickListener(new OnClickListener() {
151 
152             @Override
153             public void onClick(View v) {
154                 getFragmentManager().popBackStack();
155                 atv.changeState(ActCanvas.STATE_NORMAL);
156             }
157         });
158     }
159 }
160 
```

E.7 Leiaute Barra

Activity_BarrasTab.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.util.ArrayList;
 4
 5 import android.app.Fragment;
 6 import android.database.Cursor;
 7 import android.os.Bundle;
 8 import android.support.v4.widget.SimpleCursorAdapter;
 9 import android.util.Log;
10 import android.view.LayoutInflater;
11 import android.view.View;
12 import android.view.View.OnClickListener;
13 import android.view.ViewGroup;
14 import android.widget.AdapterView;
15 import android.widget.AdapterView.OnItemSelectedListener;
16 import android.widget.CheckBox;
17 import android.widget.CompoundButton;
18 import android.widget.CompoundButton.OnCheckedChangeListener;
19 import android.widget.ImageButton;
20 import android.widget.Spinner;
21
22 public class Activity_BarrasTab extends Fragment {
23
24     private Spinner spnMaterial;
25     private Spinner spnPerfil;
26     private Spinner spnDistribuida;
27     private ImageButton retBarras;
28     private CheckBox chbRotD;
29     private CheckBox chbRotE;
30     private DataHandler handler;
31     private ArrayList<Integer> bSelec;
32     private Cursor lMat;
33     private Cursor lPer;
34     private Cursor lDis;
35     private ActCanvas atv;
36
37     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
   savedInstanceState) {
38         View view = inflater.inflate(R.layout.activity_barratab, null);
39         atv = (ActCanvas) getActivity();
40
41         spnMaterial = (Spinner) view.findViewById(R.id.brs_spn_mat);
42         spnPerfil = (Spinner) view.findViewById(R.id.brs_spn_per);
43         spnDistribuida = (Spinner) view.findViewById(R.id.brs_spn_dis);
44         retBarras = (ImageButton) view.findViewById(R.id.brs_btn_ret);
45         chbRotD = (CheckBox) view.findViewById(R.id.brs_btn_rotD);
46         chbRotE = (CheckBox) view.findViewById(R.id.brs_btn_rotE);
47         handler = new DataHandler(atv.getBaseContext());
48         bSelec = atv.getBSelec();
49
50         handler.open();
51         lMat = handler.returnData("MATERIAIS");
52         lPer = handler.returnData("PERFIS");
53         lDis = handler.returnData("CARGASDISTRIBUIDAS");
54
55         String from[] = {"nome"};
56         int to[] = {R.id.spn_txv};
57
58         spnMaterial.setAdapter(new
   SimpleCursorAdapter(atv.getBaseContext(),R.layout.spn,lMat,from,to,0));
59         spnPerfil.setAdapter(new
```

```
   SimpleCursorAdapter(atv.getBaseContext(),R.layout.spn,lPer,from,to,0));
60         spnDistribuida.setAdapter(new
   SimpleCursorAdapter(atv.getBaseContext(),R.layout.spn,lDis,from,to,0));
61
62         createForms();
63
64         return view;
65     }
66
67     public void createForms() {
68         spnMaterial.setOnItemSelectedListener(new OnItemSelectedListener() {
69             @Override
70             public void onItemSelected(AdapterView<?> parent, View view,
71                     int position, long id) {
72                 StringBuilder query = new StringBuilder();
73                 query.append("UPDATE BARRAS SET nMat = "+position+" WHERE _id =
   "+bSelec.get(0));
74                 for(int i = 1; i < bSelec.size(); i++) {
75                     query.append(" OR _id = "+bSelec.get(i));
76                 }
77                 handler.execSQL(query.toString());
78                 atv.E.populateBarras();
79                 atv.mCanvasView.postInvalidate();
80             }
81             @Override
82             public void onNothingSelected(AdapterView<?> parent) {}
83         });
84
85         spnPerfil.setOnItemSelectedListener(new OnItemSelectedListener() {
86             @Override
87             public void onItemSelected(AdapterView<?> parent, View view,
88                     int position, long id) {
89                 StringBuilder query = new StringBuilder();
90                 query.append("UPDATE BARRAS SET nPer = "+position+" WHERE _id =
   "+bSelec.get(0));
91                 for(int i = 1; i < bSelec.size(); i++) {
92                     query.append(" OR _id = "+bSelec.get(i));
93                 }
94                 handler.execSQL(query.toString());
95                 atv.E.populateBarras();
96                 atv.mCanvasView.postInvalidate();
97             }
98             @Override
99             public void onNothingSelected(AdapterView<?> parent) {}
100        });
101
102        spnDistribuida.setOnItemSelectedListener(new OnItemSelectedListener() {
103            @Override
104            public void onItemSelected(AdapterView<?> parent, View view,
105                    int position, long id) {
106                StringBuilder query = new StringBuilder();
107                query.append("UPDATE BARRAS SET cDis = "+position+" WHERE _id =
   "+bSelec.get(0));
108                for(int i = 1; i < bSelec.size(); i++) {
109                    query.append(" OR _id = "+bSelec.get(i));
110                }
111                handler.execSQL(query.toString());
112                atv.E.populateBarras();
113                atv.mCanvasView.postInvalidate();
114                }
115            @Override
```

```
116             public void onNothingSelected(AdapterView<?> parent) {}
117         });
118 
119         chbRotE.setOnCheckedChangeListener(new OnCheckedChangeListener() {
120 
121             @Override
122             public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
123                 StringBuilder query = new StringBuilder();
124                 query.append("UPDATE BARRAS SET rotE = "+(isChecked? "1" : "0")+" WHERE _id
    = "+bSelec.get(0));
125                 for(int i = 1; i < bSelec.size(); i++) {
126                     query.append(" OR _id = "+bSelec.get(i));
127                 }
128                 handler.execSQL(query.toString());
129                 atv.E.populateBarras();
130                 atv.E.checkNosRot();
131                 atv.E.populateNos();
132                 atv.mCanvasView.postInvalidate();
133             }
134         });
135 
136         chbRotD.setOnCheckedChangeListener(new OnCheckedChangeListener() {
137 
138             @Override
139             public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
140                 StringBuilder query = new StringBuilder();
141                 query.append("UPDATE BARRAS SET rotD = "+(isChecked? "1" : "0")+" WHERE _id
    = "+bSelec.get(0));
142                 for(int i = 1; i < bSelec.size(); i++) {
143                     query.append(" OR _id = "+bSelec.get(i));
144                 }
145                 handler.execSQL(query.toString());
146                 atv.E.populateBarras();
147                 atv.E.checkNosRot();
148                 atv.E.populateNos();
149                 atv.mCanvasView.postInvalidate();
150             }
151         });
152 
153         retBarras.setOnClickListener(new OnClickListener() {
154 
155             @Override
156             public void onClick(View v) {
157                 atv.changeState(atv.STATE_NORMAL);
158                 getFragmentManager().popBackStack();
159             }
160         });
161     }
162 }
163 
```

E.8 Leiaute Grelha

Activity_GridTab.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.text.DecimalFormat;
18
19 public class Activity_GridTab extends Fragment {
20
21     private EditText edtX;
22     private EditText edtY;
23     private ImageButton btnRet;
24     private ActCanvas atv;
25     private DecimalFormat df;
26     private Ponto G;
27
28     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
   savedInstanceState) {
29         View view = inflater.inflate(R.layout.activity_gridtab, null);
30
31         edtX = (EditText) view.findViewById(R.id.grd_edt_x);
32         edtY = (EditText) view.findViewById(R.id.grd_edt_y);
33         btnRet = (ImageButton) view.findViewById(R.id.grd_btn_ret);
34         atv = (ActCanvas) getActivity();
35         df = (DecimalFormat) NumberFormat.getNumberInstance(Locale.GERMAN);
36         df.setMaximumFractionDigits(2);
37         df.setMinimumFractionDigits(2);
38
39         G = atv.getGrid();
40         edtX.setText(String.valueOf(df.format(G.getX())));
41         edtY.setText(String.valueOf(df.format(G.getY())));
42
43         edtX.addTextChangedListener(new TextWatcher() {
44
45             @Override
46             public void onTextChanged(CharSequence s, int start, int before, int count) {
47                 try {
48                     G = new Ponto(Double.parseDouble(edtX.getText().toString()),
49                                   Double.parseDouble(edtY.getText().toString()));}
50                 catch(Exception e) {}
51                 atv.setGrid(G);
52             }
53
54             @Override
55             public void beforeTextChanged(CharSequence s, int start, int count,
56                     int after) {
57             }
58
59             @Override
60             public void afterTextChanged(Editable s) {
61             }
62         });
63
64         edtY.addTextChangedListener(new TextWatcher() {
65
66             @Override
67             public void onTextChanged(CharSequence s, int start, int before, int count) {
68                 try{
69                     G = new Ponto(Double.parseDouble(edtX.getText().toString()),
70                                   Double.parseDouble(edtY.getText().toString()));}
71                 catch(Exception e) {}
72                 atv.setGrid(G);
73             }
74
```

```
75              @Override
76              public void beforeTextChanged(CharSequence s, int start, int count,
77                      int after) {
78              }
79
80              @Override
81              public void afterTextChanged(Editable s) {
82              }
83          });
84
85          btnRet.setOnClickListener(new OnClickListener() {
86
87              @Override
88              public void onClick(View v) {
89                  getFragmentManager().popBackStack();
90                  atv.changeState(atv.STATE_NORMAL);
91              }
92          });
93
94          edtX.setOnFocusChangeListener(new OnFocusChangeListener() {
95
96              @Override
97              public void onFocusChange(View v, boolean hasFocus) {
98                  if(!hasFocus) {
99                      edtX.setText(String.valueOf(G.getX()));
100                 }
101
102             }
103         });
104
105         edtY.setOnFocusChangeListener(new OnFocusChangeListener() {
106
107             @Override
108             public void onFocusChange(View v, boolean hasFocus) {
109                 if(!hasFocus) {
110                     edtY.setText(String.valueOf(G.getY()));
111                 }
112
113             }
114         });
115
116         return view;
117     }
118 }
119
```

E.9 Leiaute Principal

ActCanvas.java

```
 1 package com.tcc.ecalc;
 2
 3 import java.text.DecimalFormat;
 4 import java.util.ArrayList;
 5 import java.util.List;
 6
 7 import android.animation.ValueAnimator;
 8 import android.animation.ValueAnimator.AnimatorUpdateListener;
 9 import android.app.Fragment;
10 import android.app.FragmentManager;
11 import android.content.Context;
12 import android.content.res.Configuration;
13 import android.content.res.TypedArray;
14 import android.graphics.Canvas;
15 import android.graphics.Color;
16 import android.graphics.Paint;
17 import android.graphics.Paint.Align;
18 import android.graphics.Paint.Style;
19 import android.graphics.Path;
20 import android.graphics.Path.Direction;
21 import android.graphics.Point;
22 import android.graphics.RectF;
23 import android.os.Bundle;
24 import android.support.v4.app.ActionBarDrawerToggle;
25 import android.support.v4.app.FragmentActivity;
26 import android.support.v4.widget.DrawerLayout;
27 import android.util.Log;
28 import android.view.GestureDetector;
29 import android.view.Gravity;
30 import android.view.KeyEvent;
31 import android.view.Menu;
32 import android.view.MenuItem;
33 import android.view.MotionEvent;
34 import android.view.ScaleGestureDetector;
35 import android.view.View;
36 import android.view.View.OnClickListener;
37 import android.view.WindowManager;
38 import android.view.animation.OvershootInterpolator;
39 import android.widget.AdapterView;
40 import android.widget.FrameLayout;
41 import android.widget.ImageButton;
42 import android.widget.LinearLayout;
43 import android.widget.ListView;
44 import android.widget.SeekBar;
45 import android.widget.SeekBar.OnSeekBarChangeListener;
46 import android.widget.Toast;
47
48 import com.tcc.ecalc.IntegerStore.IntListener;
49 import com.tcc.ecalc.SwapButton.onSwapListener;
50
51 public class ActCanvas extends FragmentActivity {
52
53     //Objeto que reterá todos os dados da E
54     Estrutura E;
55
56     //Formatador de texto
57     private DecimalFormat format = new DecimalFormat("0.00");
58
59     //Variáveis da view
60     Ponto mGrid = new Ponto(1,1);
61     Ponto mOrg = new Ponto(0,0);
```

```
62      Ponto mSize;
63      double mEscala = 50;
64      int moveMode = 0;
65
66      IntegerStore mState;
67      int mSelec;
68      int mResultados;
69      int fragInformation;
70
71      //Objetos de layout
72      ImageButton mButton1;
73      ImageButton mButton2;
74      ImageButton mButton3;
75      ImageButton mButtonGrid;
76      SwapButton mSwapRes;
77
78      VerticalSeekBar mSeekBar;
79      LinearLayout mCanvas;
80      View_Canvas mCanvasView;
81      FrameLayout fragContainer;
82      float densidade;
83
84      Ponto P0 = new Ponto();
85      Ponto P1 = new Ponto();
86
87      Ponto b1 = new Ponto();
88      Ponto b2 = new Ponto();
89      //MENU
90
91
92      //Recebe o layout de gaveta
93      private DrawerLayout mDrawerLayout;
94      //Recebe a lista de menu
95      private ListView mDrawerList;
96      //Nome dos itens do menu
97      private String[] navMenuTitles;
98      //Imagens dos itens do menu
99      private TypedArray navMenuIcons;
100     //Nome dos itens
101     private ArrayList<NavDrawerItem> navDrawerItems;
102     //Adaptador do menu
103     private NavDrawerListAdapter adapter;
104     //Toggle do menu
105     private ActionBarDrawerToggle mDrawerToggle;
106
107     public final static int STATE_NORMAL = 0;
108     public final static int STATE_MENU = 1;
109     public final static int STATE_SELECIONABARRA = 1;
110     public final static int STATE_SELECIONANO = 2;
111     public final static int STATE_ADICIONABARRA1 = 3;
112     public final static int STATE_ADICIONABARRA2 = 4;
113     public final static int STATE_FRAGMENT = 6;
114     public final static int STATE_RESULTADOS = 7;
115
116     public final static int SELEC_NORMAL = 0;
117     public final static int SELEC_SETAPOIO = 1;
118     public final static int SELEC_SETPONTUAL = 2;
119     public final static int SELEC_SETDISTRIBUIDA = 3;
120     public final static int SELEC_SETPERFIL = 4;
121     public final static int SELEC_SETMATERIAL = 5;
122
```

```
123     public final static int RES_REACOES = 0;
124     public final static int RES_DESLOCAMENTOS = 1;
125     public final static int RES_NORMAL = 2;
126     public final static int RES_CORTANTE = 3;
127     public final static int RES_MOMENTO = 4;
128
129
130     public final static int ANIMATION_DURATION = 1000;
131
132     public final static int DISTANCIAMAXIMA = 50;
133
134     protected void onCreate(Bundle savedInstanceState) {
135
136         super.onCreate(savedInstanceState);
137         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
138                 WindowManager.LayoutParams.FLAG_FULLSCREEN);
139         setContentView(R.layout.canvas);
140
141         E = new Estrutura("Estrutura.db", getBaseContext());
142
143     //Identifica o tamanho da tela
144
145         Point p = new Point();
146         getWindowManager().getDefaultDisplay().getSize(p);
147         densidade = getBaseContext().getResources().getDisplayMetrics().density;
148         mSize = new Ponto((double) p.x,(double) p.y);
149
150         mState = new IntegerStore(STATE_NORMAL);
151         mState.setListener(new IntListener() {
152
153             @Override
154             public void onValueChanged(int value) {
155                 Log.d("Ecalc","mState = "+mState.getValue());
156                 switch(mState.getValue()) {
157                 case STATE_NORMAL : {
158                     for(int i = 0; i < E.mBarras.length; i++) {
159                         E.mBarras[i].setSelection(false);
160                     }
161                     for(int i = 0; i < E.mNos.length; i++) {
162                         E.mNos[i].setSelection(false);
163                     }
164                     mCanvasView.postInvalidate();
165                     mButton1.setVisibility(View.VISIBLE);
166                     mButton2.setVisibility(View.VISIBLE);
167                     mButton3.setVisibility(View.GONE);
168                     mSwapRes.setVisibility(View.GONE);
169                     mSeekBar.setVisibility(View.GONE);
170                     mButton1.setBackgroundResource(R.drawable.cog);
171                     mButton2.setBackgroundResource(R.drawable.ad_barra);
172                     break;
173                     }
174                 case STATE_SELECIONANO : {
175                     mSwapRes.setVisibility(View.GONE);
176                     mSeekBar.setVisibility(View.GONE);
177                     mButton1.setVisibility(View.VISIBLE);
178                     mButton2.setVisibility(View.VISIBLE);
179                     if(mSelec == SELEC_NORMAL) {
180                         mButton3.setVisibility(View.VISIBLE);
181                         mButton1.setBackgroundResource(R.drawable.cog);
182                         mButton2.setBackgroundResource(R.drawable.excluir);
183                         mButton3.setBackgroundResource(R.drawable.retrn);
```

```
184                     } else {
185                         mButton3.setVisibility(View.GONE);
186                         mButton1.setBackgroundResource(R.drawable.tick);
187                         mButton2.setBackgroundResource(R.drawable.retrn);
188                         }
189                     break;
190                     }
191                 case STATE_SELECIONABARRA : {
192                     mSwapRes.setVisibility(View.GONE);
193                     mSeekBar.setVisibility(View.GONE);
194                     mButton1.setVisibility(View.VISIBLE);
195                     mButton2.setVisibility(View.VISIBLE);
196                     if(mSelec == SELEC_NORMAL) {
197                         mButton3.setVisibility(View.VISIBLE);
198                         mButton1.setBackgroundResource(R.drawable.cog);
199                         mButton2.setBackgroundResource(R.drawable.excluir);
200                         mButton3.setBackgroundResource(R.drawable.retrn);
201                     } else {
202                         mButton3.setVisibility(View.GONE);
203                         mButton1.setBackgroundResource(R.drawable.tick);
204                         mButton2.setBackgroundResource(R.drawable.retrn);
205                         }
206                     break;
207                     }
208                 case STATE_ADICIONABARRA1 : {
209                     mSwapRes.setVisibility(View.GONE);
210                     mSeekBar.setVisibility(View.GONE);
211                     mButton1.setVisibility(View.VISIBLE);
212                     mButton1.setBackgroundResource(R.drawable.retrn);
213                     mButton2.setVisibility(View.GONE);
214                     mButton3.setVisibility(View.GONE);
215                     break;
216                     }
217                 case STATE_ADICIONABARRA2 : {
218                     mSwapRes.setVisibility(View.GONE);
219                     mSeekBar.setVisibility(View.GONE);
220                     mButton1.setVisibility(View.VISIBLE);
221                     mButton1.setBackgroundResource(R.drawable.retrn);
222                     mButton2.setVisibility(View.GONE);
223                     mButton3.setVisibility(View.GONE);
224                     break;
225                     }
226                 case STATE_FRAGMENT : {
227                     mSwapRes.setVisibility(View.GONE);
228                     mSeekBar.setVisibility(View.GONE);
229                     mButton1.setVisibility(View.GONE);
230                     mButton2.setVisibility(View.GONE);
231                     mButton3.setVisibility(View.GONE);
232                     break;
233                 }
234                 case STATE_RESULTADOS : {
235                     mSwapRes.setVisibility(View.VISIBLE);
236                     mSeekBar.setVisibility(View.VISIBLE);
237                     mButton1.setVisibility(View.VISIBLE);
238                     mButton2.setVisibility(View.GONE);
239                     mButton3.setVisibility(View.GONE);
240                     break;
241                     }
242                 }
243             }
244         });
```

```
245
246
247         mCanvasView = new View_Canvas(getBaseContext());
248         mCanvas = (LinearLayout) findViewById(R.id.canvas);
249
250         fragContainer = (FrameLayout) findViewById(R.id.frame_container);
251         mButton1 = (ImageButton) findViewById(R.id.btn_1);
252         mButton2 = (ImageButton) findViewById(R.id.btn_2);
253         mButton3 = (ImageButton) findViewById(R.id.btn_3);
254         mButtonGrid = (ImageButton) findViewById(R.id.btn_grid);
255
256         mSeekBar = (VerticalSeekBar) findViewById(R.id.seekbar);
257         mSeekBar.setMax(200);
258         mSeekBar.setProgress(100);
259         mSeekBar.setEnabled(true);
260
261         mSwapRes = (SwapButton) findViewById(R.id.swapres);
262         mSwapRes.addItem(R.drawable.btn_esfreacoes);
263         mSwapRes.addItem(R.drawable.btn_esfdelocamentos);
264         mSwapRes.addItem(R.drawable.btn_esfcortante);
265         mSwapRes.addItem(R.drawable.btn_esfnormal);
266         mSwapRes.addItem(R.drawable.btn_esfmomento);
267
268         //Cria a view
269         mCanvas.addView(mCanvasView);
270
271         mState.setValue(STATE_NORMAL);
272
273         mButtonGrid.setOnClickListener(new OnClickListener() {
274
275             @Override
276             public void onClick(View v) {
277                 if(mState.getValue() != STATE_FRAGMENT) {
278                     Fragment fragment = new Activity_GridTab();
279                     fragContainer.getLayoutParams().width = (int) (150*densidade);
280                     mState.setValue(STATE_FRAGMENT);
281                     FragmentManager fragmentManager = getFragmentManager();
282                     fragmentManager.beginTransaction()
283                         .add(R.id.frame_container,
    fragment).addToBackStack(null).commit();
284                 }
285             }
286         });
287
288         mSwapRes.setOnSwapListener(new onSwapListener() {
289
290             @Override
291             public void onSwap(int position) {
292                 switch(position) {
293                 case RES_REACOES :
294                     mResultados = RES_REACOES;
295                     break;
296                 case RES_DESLOCAMENTOS :
297                     mResultados = RES_DESLOCAMENTOS;
298                     break;
299                 case RES_NORMAL :
300                     mResultados = RES_NORMAL;
301                     break;
302                 case RES_CORTANTE :
303                     mResultados = RES_CORTANTE;
304                     break;
```

```
305                 case RES_MOMENTO :
306                     mResultados = RES_MOMENTO;
307                     break;
308                 }
309                 mCanvasView.postInvalidate();
310             }
311         });
312 
313         mSeekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
314 
315             @Override
316             public void onStopTrackingTouch(SeekBar seekBar) {}
317 
318             @Override
319             public void onStartTrackingTouch(SeekBar seekBar) {}
320 
321             @Override
322             public void onProgressChanged(SeekBar seekBar, int progress,
323                     boolean fromUser) {
324                 mCanvasView.postInvalidate();
325             }
326         });
327 
328         mButton1.setOnClickListener(new OnClickListener() {
329 
330             @Override
331             public void onClick(View v) {
332                 switch(mState.getValue()) {
333                 case STATE_NORMAL :
334                     if(mDrawerLayout.isDrawerOpen(Gravity.START)){
335                         mDrawerLayout.closeDrawer(Gravity.START);} else
336                     {mDrawerLayout.openDrawer(Gravity.START);}
337                     break;
338                 case STATE_ADICIONABARRA1 :
339                     mState.setValue(STATE_NORMAL);
340                     break;
341                 case STATE_ADICIONABARRA2 :
342                     mState.setValue(STATE_NORMAL);
343                     break;
344                 case STATE_SELECIONABARRA :
345                     switch(mSelec) {
346                     case SELEC_NORMAL :
347                         Fragment fragment = new Activity_BarrasTab();
348                         fragContainer.getLayoutParams().width = (int) (150*densidade);
349                         FragmentManager fragmentManager = getFragmentManager();
350                         fragmentManager.beginTransaction()
351                                 .add(R.id.frame_container,
    fragment).addToBackStack(null).commit();
352                         mState.setValue(STATE_FRAGMENT);
353                         break;
354                     case SELEC_SETDISTRIBUIDA : {
355                         ArrayList<Integer> selec = new ArrayList<Integer>();
356                         for(int i = 0; i < E.mBarras.length; i++) {
357                             if(E.mBarras[i].getSelection()) {
358                                 selec.add(i);
359                             }
360                         }
361                         if(selec.size() > 0) {
362                             E.setDistribuida(selec,fragInformation);
363                         }
364                         mState.setValue(STATE_NORMAL);
```

```
365                         mSelec = SELEC_NORMAL;
366                         mCanvasView.postInvalidate();
367                         break;
368                     }
369                     case SELEC_SETPERFIL : {
370                         ArrayList<Integer> selec = new ArrayList<Integer>();
371                         for(int i = 0; i < E.mBarras.length; i++) {
372                             if(E.mBarras[i].getSelection()) {
373                                 selec.add(i);
374                             }
375                         }
376                         if(selec.size() > 0) {
377                             E.setPerfil(selec,fragInformation);
378                         }
379                         mState.setValue(STATE_NORMAL);
380                         mSelec = SELEC_NORMAL;
381                         mCanvasView.postInvalidate();
382                         break;
383                     }
384                     case SELEC_SETMATERIAL : {
385                         ArrayList<Integer> selec = new ArrayList<Integer>();
386                         for(int i = 0; i < E.mBarras.length; i++) {
387                             if(E.mBarras[i].getSelection()) {
388                                 selec.add(i);
389                             }
390                         }
391                         if(selec.size() > 0) {
392                             E.setMaterial(selec, fragInformation);
393                         }
394                         mState.setValue(STATE_NORMAL);
395                         mSelec = SELEC_NORMAL;
396                         mCanvasView.postInvalidate();
397                         break;
398                     }
399                     }
400                     break;
401                 case STATE_SELECIONANO :
402                     switch(mSelec) {
403                     case SELEC_NORMAL :
404                         Fragment fragment = new Activity_NosTab();
405                         fragContainer.getLayoutParams().width = (int) (150*densidade);
406                         FragmentManager fragmentManager = getFragmentManager();
407                         fragmentManager.beginTransaction()
408                                 .add(R.id.frame_container,
    fragment).addToBackStack(null).commit();
409                         mState.setValue(STATE_FRAGMENT);
410                         break;
411                     case SELEC_SETAPOIO : {
412                         ArrayList<Integer> selec = new ArrayList<Integer>();
413                         for(int i = 0; i < E.mNos.length; i++) {
414                             if(E.mNos[i].getSelection()) {
415                                 selec.add(i);
416                             }
417                         }
418                         if(selec.size() > 0) {
419                             E.setApoio(selec, fragInformation);
420                         }
421                         mState.setValue(STATE_NORMAL);
422                         mSelec = SELEC_NORMAL;
423                         mCanvasView.postInvalidate();
424                         break;
```

```
425                     }
426                     case SELEC_SETPONTUAL : {
427                         ArrayList<Integer> selec = new ArrayList<Integer>();
428                         for(int i = 0; i < E.mNos.length; i++) {
429                             if(E.mNos[i].getSelection()) {
430                                 selec.add(i);
431                             }
432                         }
433                         if(selec.size() > 0) {
434                             E.setPontual(selec,fragInformation);
435                         }
436                         mState.setValue(STATE_NORMAL);
437                         mSelec = SELEC_NORMAL;
438                         mCanvasView.postInvalidate();
439                         break;
440                     }
441                     }
442                     break;
443                 case STATE_RESULTADOS : {
444                     if(mDrawerLayout.isDrawerOpen(Gravity.START)){
445                         mDrawerLayout.closeDrawer(Gravity.START);} else
446                     {mDrawerLayout.openDrawer(Gravity.START);}
447                     mState.setValue(STATE_NORMAL);
448                     break;
449                     }
450                 }
451             }
452         });
453 
454         mButton2.setOnClickListener(new OnClickListener() {
455 
456             @Override
457             public void onClick(View v) {
458                 switch(mState.getValue()) {
459                 case STATE_NORMAL : {
460                     mState.setValue(STATE_ADICIONABARRA1);
461                     break;
462                 }
463                 case STATE_SELECIONANO : {
464                     if(mSelec == SELEC_NORMAL) {
465                         List<Integer> iIds = new ArrayList<Integer>();
466                         for(int i = 0; i < E.mNos.length; i++) {
467                             if(E.mNos[i].getSelection()) {
468                                 iIds.add(E.mNos[i].getId());
469                             }
470                         }
471                         if(iIds.size() != 0) {
472                             E.excNo(iIds);
473                         }
474                         mCanvasView.postInvalidate();
475                         mState.setValue(STATE_NORMAL);
476                     } else
477                     {
478                         mState.setValue(STATE_NORMAL);
479                         mCanvasView.postInvalidate();
480                     }
481                     break;
482                 }
483                 case STATE_SELECIONABARRA : {
484                     if(mSelec == SELEC_NORMAL) {
485                         List<Integer> iIds = new ArrayList<Integer> ();
```

```
                        for(int i = 0; i < E.mBarras.length; i++) {
                            if(E.mBarras[i].getSelection()) {
                                iIds.add(i);
                            }
                        }
                        if(iIds.size() != 0) {
                            E.excBarra(iIds);
                        }
                        mCanvasView.postInvalidate();
                        mState.setValue(STATE_NORMAL);
                    } else
                    {
                        mState.setValue(STATE_NORMAL);
                        mCanvasView.postInvalidate();
                    }
                    break;
                }
                }
            }
        });

        mButton3.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                mState.setValue(STATE_NORMAL);
                for(int i = 0; i < E.mBarras.length; i++) {
                    E.mBarras[i].setSelection(false);
                }
                for(int i = 0; i < E.mNos.length; i++) {
                    E.mNos[i].setSelection(false);
                }
            }
        });
        //Carrega os itens
        navMenuTitles = getResources().getStringArray(R.array.nav_drawer_items);
        //Carrega os icones
        navMenuIcons = getResources().obtainTypedArray(R.array.nav_drawer_icons);
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mDrawerList = (ListView) findViewById(R.id.list_slidermenu);
        mDrawerList.getLayoutParams().width = (int) (200*densidade);
        navDrawerItems = new ArrayList<NavDrawerItem>();

        //Adiciona todos os itens
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[0],
    navMenuIcons.getResourceId(0, -1), true, String.valueOf(E.mCDis.length-1)));
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[1],
    navMenuIcons.getResourceId(1, -1), true, String.valueOf(E.mCPon.length-1)));
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[2],
    navMenuIcons.getResourceId(2, -1)));
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[3],
    navMenuIcons.getResourceId(3, -1), true, String.valueOf(E.mPerfis.length-1)));
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[4],
    navMenuIcons.getResourceId(4, -1), true, String.valueOf(E.mMateriais.length-1)));
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[5],
    navMenuIcons.getResourceId(4, -1)));
        navDrawerItems.add(new NavDrawerItem(navMenuTitles[6],
    navMenuIcons.getResourceId(4, -1)));

        //Recicla a array
        navMenuIcons.recycle();
```

```
540         //Designa o adpater para o menu
541         adapter = new NavDrawerListAdapter(getApplicationContext(), navDrawerItems);
542         mDrawerList.setAdapter(adapter);
543         mDrawerList.setOnItemClickListener(new SlideMenuClickListener());
544         //Aciona o menu
545         mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
546                 R.drawable.per_icn_1, //nav menu toggle icon
547                 R.string.app_name, // nav drawer open - description for accessibility
548                 R.string.app_name // nav drawer close - description for accessibility
549         ){
550             public void onDrawerClosed(View view) {
551                 invalidateOptionsMenu();
552             }
553 
554             public void onDrawerOpened(View drawerView) {
555                 invalidateOptionsMenu();
556             }
557         };
558         mDrawerLayout.setDrawerListener(mDrawerToggle);
559 
560     }
561 
562     public class View_Canvas extends View {
563 
564         private Ponto p = new Ponto();
565         private Ponto q = new Ponto();
566         private Ponto[] X = new Ponto[]{new Ponto(),new Ponto(), new Ponto()};
567         private ScaleGestureDetector scaleDetector;
568         private GestureDetector gestureDetector;
569         private CargaPontual cPon;
570         private CargaDistribuida cDist;
571         private Path path;
572         private int menorPonto;
573         private BigMatrix resLocal;
574 
575         public final float MOVE_DISTANCIA = 1;
576         public final int MOVE_NORMAL = 0;
577         public final int MOVE_DRAG = 1;
578         public final int MOVE_ZOOM = 2;
579 
580         public final int DRAW_POINTSIZE = 10;
581 
582         float x, y, nP = 40;
583 
584         Paint mPaintApoio = new Paint();
585         Paint mPaintNo = new Paint();
586         Paint mPaintGrid = new Paint();
587         Paint mPaintPontual = new Paint();
588         Paint mPaintBarra = new Paint();
589         Paint mPaintDistribuida = new Paint();
590         Paint mPaintTexto = new Paint();
591         Paint mEraser = new Paint();
592 
593         public View_Canvas(Context context){
594             super(context);
595 
596             path = new Path();
597 
598             scaleDetector = new ScaleGestureDetector(context, new ScaleListener());
599             gestureDetector = new GestureDetector(context, new GestureListener());
600 
```

```
601             mPaintNo.setColor(getResources().getColor(R.color.No));
602             mPaintNo.setStrokeWidth(3f);
603             mPaintNo.setStyle(Paint.Style.STROKE);
604 
605             //Características de pintura Grid
606             mPaintGrid.setColor(getResources().getColor(R.color.Grid));
607             mPaintGrid.setStrokeWidth(2f*densidade);
608 
609             //Características de pintura Barras
610             mPaintBarra.setColor(getResources().getColor(R.color.BarraNormal));
611             mPaintBarra.setStrokeWidth(10f);
612             mPaintBarra.setStrokeCap(Paint.Cap.ROUND);
613             mPaintBarra.setAlpha(200);
614             mPaintBarra.setStyle(Paint.Style.STROKE);
615 
616             //Características de pintura Cargas Pontuais
617             mPaintPontual.setColor(getResources().getColor(R.color.CargaPontual));
618             mPaintPontual.setStrokeWidth(2f);
619             mPaintPontual.setStyle(Paint.Style.STROKE);
620 
621             //Características de pintura Cargas Distribuidas
622             mPaintDistribuida.setColor(getResources().getColor(R.color.CargaDistribuida));
623             mPaintDistribuida.setStrokeWidth(2f);
624             mPaintDistribuida.setStyle(Paint.Style.STROKE);
625 
626             //Características de pintura Texto
627             mPaintTexto.setColor(getResources().getColor(R.color.Texto));
628             mPaintTexto.setStrokeWidth(1f);
629             mPaintTexto.setTextSize(10*densidade);
630 
631             //Características de pintura Apoios
632             mPaintApoio.setColor(getResources().getColor(R.color.Apoio));
633             mPaintApoio.setStrokeWidth(2f);
634             mPaintApoio.setStyle(Paint.Style.STROKE);
635 
636             //Borracha
637             mEraser.setColor(Color.WHITE);
638         }
639 
640         @Override
641         public boolean onKeyDown(int keyCode, android.view.KeyEvent event) {
642             if(event.getAction() == KeyEvent.ACTION_DOWN) {
643                 if(keyCode == KeyEvent.KEYCODE_BACK) {
644                     getFragmentManager().popBackStack();
645                     return true;
646                 }
647             }
648             return false;
649         }
650 
651         @Override
652         public boolean onTouchEvent(MotionEvent event) {
653             switch(event.getAction() & MotionEvent.ACTION_MASK) {
654             case MotionEvent.ACTION_DOWN :
655                 moveMode = MOVE_DRAG;
656                 X[0].setX(event.getX());
657                 X[0].setY(event.getY());
658                 X[1] = new Ponto(X[0]);
659                 X[2] = new Ponto(X[0]);
660                 switch(mState.getValue()) {
661                 case STATE_SELECIONABARRA :
```

```
662                     if(getBarraSelecionada(X[1])) {
663                         for(int i = 0; i < E.mBarras.length; i++) {
664                             if(E.mBarras[i].getSelection()) {
665                                 break;
666                             } else {
667                                 if(i == E.mBarras.length-1) {
668                                     if(mSelec == SELEC_NORMAL) {
669                                         mState.setValue(STATE_NORMAL);
670                                     }
671                                 }
672                             }
673                         }
674                     }
675                     invalidate();
676                     break;
677                 case STATE_SELECIONANO :
678                     if(getNoSelecionado(X[1])) {
679                         for(int i = 0; i < E.mNos.length; i++) {
680                             if(E.mNos[i].getSelection()) {
681                                 break;
682                             } else {
683                                 if(i == E.mNos.length-1) {
684                                     if(mSelec == SELEC_NORMAL) {
685                                         mState.setValue(STATE_NORMAL);
686                                     }
687                                 }
688                             }
689                         }
690                     }
691                     invalidate();
692                     break;
693                 }
694                 break;
695             case MotionEvent.ACTION_MOVE :
696                 X[2] = new Ponto(event.getX(),event.getY());
697                 Ponto dX = X[2].subtrai(X[1]);
698                 dX.setY(-dX.getY());
699                 mOrg = mOrg.subtrai(dX.div(mEscala));
700                 X[1] = new Ponto(X[2]);
701                 invalidate();
702                 break;
703             case MotionEvent.ACTION_POINTER_DOWN :
704                 moveMode = MOVE_ZOOM;
705                 break;
706             case MotionEvent.ACTION_POINTER_UP :
707                 moveMode = MOVE_DRAG;
708                 break;
709             case MotionEvent.ACTION_UP :
710                 switch(mState.getValue()) {
711                 case STATE_ADICIONABARRA1 : {
712                     if(X[1].getDistancia(X[0]) <= DISTANCIAMAXIMA) {
713                         b1 = new Ponto(event.getX(),event.getY());
714                         mState.setValue(STATE_ADICIONABARRA2);
715                     }
716                     break;
717                 }
718                 case STATE_ADICIONABARRA2 : {
719                     if(X[1].getDistancia(X[0]) <= DISTANCIAMAXIMA) {
720                         b2 = new Ponto(event.getX(),event.getY());
721                         mState.setValue(STATE_ADICIONABARRA1);
722                         b1 = telaToGrid(b1);
```

```
723                         b2 = telaToGrid(b2);
724                         b1.setX(Math.round(b1.getX()/mGrid.getX())*mGrid.getX());
725                         b1.setY(Math.round(b1.getY()/mGrid.getY())*mGrid.getY());
726                         b2.setX(Math.round(b2.getX()/mGrid.getX())*mGrid.getX());
727                         b2.setY(Math.round(b2.getY()/mGrid.getY())*mGrid.getY());
728                         E.addBarra(b1, b2);
729                         E.reindex();
730                         invalidate();
731                         }
732                     break;
733                     }
734                 }
735                 moveMode = MOVE_NORMAL;
736                 break;
737             }
738 
739             scaleDetector.onTouchEvent(event);
740             gestureDetector.onTouchEvent(event);
741             return true;
742         }
743 
744         protected void onDraw(Canvas canvas) {
745             //TODO PUT IN PERSPECTIVE ACCORDING TO MAXIMUM LOADING
746             canvas.drawColor(Color.WHITE);
747             canvas.drawText("mState = "+mState.getValue(), 0, 12f, mPaintTexto);
748             if((mEscala*mGrid.getX() >= 30) && (mEscala*mGrid.getY() >= 30)) {
749                 double i0 = (Math.round((mOrg.getX() -
    0.5*mSize.getX()/mEscala)/mGrid.getX()))*mGrid.getX();
750                 double j0 = (Math.round((mOrg.getY() -
    0.5*mSize.getY()/mEscala)/mGrid.getY()))*mGrid.getY();
751                 double i1 = (Math.round((mOrg.getX() +
    0.5*mSize.getX()/mEscala)/mGrid.getX()))*mGrid.getX();
752                 double j1 = (Math.round((mOrg.getY() +
    0.5*mSize.getY()/mEscala)/mGrid.getY()))*mGrid.getY();
753                 for(double i = i0; i <= i1; i += mGrid.getX()) {
754                     for(double j = j0; j <= j1; j += mGrid.getY()) {
755                         p.setX(i);
756                         p.setY(j);
757                         p = gridToTela(p);
758                         canvas.drawPoint((float) p.getX(), (float) p.getY(), mPaintGrid);
759                     }
760                 }
761             }
762             if(mState.getValue() != STATE_RESULTADOS) {
763                 for(int i = 0; i < E.mBarras.length; i++) {
764                     p = (E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)]).getC();
765                     q = (E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)]).getC();
766                     //TODO VERIFICAR ESTA CONDIÇÃO
767                     if(E.mBarras[i].getAngulo() > -180 && E.mBarras[i].getAngulo() <= 180)
    {
768                         if(p.getX() < q.getX()) {
769                             menorPonto = 0;
770                         } else {
771                             menorPonto = 1;
772                         }
773                     } else {
774                         if(p.getX() < q.getX()) {
775                             menorPonto = 1;
776                         } else {
777                             menorPonto = 0;
778                         }
```

```
779                     }
780                     p = gridToTela(p);
781                     q = gridToTela(q);
782                     mPaintNo.setStyle(Style.STROKE);
783                     if(E.mBarras[i].getQ() != 0) {
784                         cDist = E.mCDis[E.mBarras[i].getQ()];
785                         if(cDist.isGlobal()) {
786                             if(cDist.getQ(0) != 0 || cDist.getQ(1) != 0 || cDist.getQ(2) !
    = 0 || cDist.getQ(3) != 0) {
787                                 path.reset();
788                                 switch (menorPonto) {
789                                 case 0 :
790                                     path.moveTo((float) p.getX(),(float) p.getY());
791                                     path.lineTo((float) p.getX(),(float) (p.getY() +
    cDist.getQ(1)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala));
792                                     path.lineTo((float) q.getX(),(float) (q.getY() +
    cDist.getQ(3)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala));
793                                     path.lineTo((float) q.getX(),(float) q.getY());
794                                     canvas.drawText(format.format(cDist.getQ(1))+"kN/m",(f
    loat) p.getX(),(float) (p.getY()+
    1.05*cDist.getQ(1)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala),
    mPaintTexto);
795                                     canvas.drawText(format.format(cDist.getQ(3))+"kN/m",(f
    loat) q.getX(),(float) (q.getY()+
    1.05*cDist.getQ(3)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala),
    mPaintTexto);
796                                     canvas.drawText(format.format(cDist.getQ(0))+"kN/m",(f
    loat) p.getX()+10,(float) (p.getY() - 10), mPaintTexto);
797                                     canvas.drawText(format.format(cDist.getQ(2))+"kN/m",(f
    loat) q.getX()-30,(float) (q.getY() - 10), mPaintTexto);
798                                 break;
799                                 case 1 :
800                                     path.moveTo((float) p.getX(),(float) p.getY());
801                                     path.lineTo((float) p.getX(),(float) (p.getY() +
    cDist.getQ(1)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala));
802                                     path.lineTo((float) q.getX(),(float) (q.getY() +
    cDist.getQ(3)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala));
803                                     path.lineTo((float) q.getX(),(float) q.getY());
804                                     canvas.drawText(format.format(cDist.getQ(1))+"kN/m",(f
    loat) q.getX()+5,(float) (q.getY() +
    1.05*cDist.getQ(1)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala),
    mPaintTexto);
805                                     canvas.drawText(format.format(cDist.getQ(3))+"kN/m",(f
    loat) p.getX()+5,(float) (p.getY() +
    1.05*cDist.getQ(3)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala),
    mPaintTexto);
806                                     canvas.drawText(format.format(cDist.getQ(2))+"kN/m",(f
    loat) p.getX()+10,(float) (p.getY() - 10), mPaintTexto);
807                                     canvas.drawText(format.format(cDist.getQ(0))+"kN/m",(f
    loat) q.getX()-30,(float) (q.getY() - 10), mPaintTexto);
808                                     break;
809                                 }
810                                 canvas.drawPath(path, mPaintDistribuida);
811                             }
812                         } else {
813                             if(cDist.getQ(0) != 0 || cDist.getQ(1) != 0 || cDist.getQ(2) !
    = 0 || cDist.getQ(3) != 0) {
814                                 canvas.save();
815                                 canvas.rotate((float) -E.mBarras[i].getAngulo(),(float)
    p.getX(),(float) p.getY());
816                                 path.reset();
```

```
817                                 path.moveTo((float) p.getX(), (float) p.getY());
818                                 path.lineTo((float) p.getX(),(float)
    (p.getY()+cDist.getQ(1)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))
    *mEscala));
819                                 path.lineTo((float) (p.getX()+q.getDistancia(p)),(float)
    (p.getY()+cDist.getQ(3)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))
    *mEscala));
820                                 path.lineTo((float) (p.getX()+q.getDistancia(p)),(float)
    p.getY());
821                                 mPaintTexto.setTextAlign(Align.LEFT);
822                             }
823                             if(cDist.getQ(1) != 0) {
824                                 canvas.drawText(format.format(cDist.getQ(1))+"KN/m",(float)
    p.getX(),
825                                         (float) ((p.getY()+(cDist.getQ(1)
    *1.05)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala)), mPaintTexto);
826                             }
827                             if(cDist.getQ(0) != 0) {
828                                 canvas.drawText(format.format(cDist.getQ(0))+"KN/m",(float)
    p.getX(),(float) (p.getY()-10), mPaintTexto);
829                             }
830                             mPaintTexto.setTextAlign(Align.RIGHT);
831                             if(cDist.getQ(3) != 0) {
832                                 canvas.drawText(format.format(cDist.getQ(3))+"KN/m",(float)
    (p.getX()+q.getDistancia(p)),
833                                     (float) ((p.getY()+(cDist.getQ(3)
    *1.05)/Math.max(Math.abs(cDist.getQ(1)),Math.abs(cDist.getQ(3)))*mEscala)), mPaintTexto);
834                             }
835                             if(cDist.getQ(2) != 0) {
836                                 canvas.drawText(format.format(cDist.getQ(2))+"KN/m",(float)
    (p.getX()+q.getDistancia(p)),(float) (p.getY()-10), mPaintTexto);
837                             }
838                             canvas.drawPath(path, mPaintDistribuida);
839                             canvas.restore();
840                         }
841                     }
842                     if(E.mBarras[i].getSelection()) {
843                         mPaintBarra.setColor(getResources().getColor(R.color.BarraSelecion
    ada));
844                     } else {
845                         mPaintBarra.setColor(getResources().getColor(R.color.BarraNormal));
846                     }
847                     canvas.drawLine((float) p.getX(),(float) p.getY(),(float)
    q.getX(),(float) q.getY(), mPaintBarra);
848                     if(E.mBarras[i].getRot(Barra.NO_ESQUERDO) &&
    !E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)].getRot()) {
849                         if(E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)].getSelection()) {
850                             mPaintNo.setColor(getResources().getColor(R.color.BarraSelecio
    nada));
851                         } else {
852                             mPaintNo.setColor(getResources().getColor(R.color.No));
853                         }
854                         p.setX(p.getX()+DRAW_POINTSIZE*Math.cos(Math.toRadians(E.mBarras[i]
    .getAngulo())));
855                         p.setY(p.getY()-DRAW_POINTSIZE*Math.sin(Math.toRadians(E.mBarras[i]
    .getAngulo())));
856                         canvas.drawCircle((float) p.getX(),(float) p.getY(),
    DRAW_POINTSIZE, mEraser);
857                         canvas.drawCircle((float) p.getX(),(float) p.getY(),
    DRAW_POINTSIZE, mPaintNo);
858                     }
```

```
859
860                     if(E.mBarras[i].getRot(Barra.NO_DIREITO) &&
    !E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)].getRot()) {
861                         if(E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)].getSelection()) {
862                             mPaintNo.setColor(getResources().getColor(R.color.BarraSelecio
    nada));
863                         } else {
864                         mPaintNo.setColor(getResources().getColor(R.color.No));
865                         }
866                         q.setX(q.getX()-DRAW_POINTSIZE*Math.cos(Math.toRadians(E.mBarras[i]
    .getAngulo())));
867                         q.setY(q.getY()+DRAW_POINTSIZE*Math.sin(Math.toRadians(E.mBarras[i]
    .getAngulo())));
868                         canvas.drawCircle((float) q.getX(),(float) q.getY(),
    DRAW_POINTSIZE, mEraser);
869                         canvas.drawCircle((float) q.getX(),(float) q.getY(),
    DRAW_POINTSIZE, mPaintNo);
870                     }
871                 }
872
873                 for(int i = 0; i < E.mNos.length; i++) {
874                     p = E.mNos[i].getC();
875                     p = gridToTela(p);
876                     canvas.drawCircle((float) p.getX(),(float)
    p.getY(),DRAW_POINTSIZE/2,mEraser);
877                     if(E.mNos[i].getSelection()) {
878                         mPaintNo.setColor(getResources().getColor(R.color.BarraSelecionada)
    );
879                     } else {
880                         mPaintNo.setColor(getResources().getColor(R.color.No));
881                     }
882                     if(E.mNos[i].getRot()) {
883                         mPaintNo.setStyle(Style.STROKE);
884                     } else {
885                         mPaintNo.setStyle(Style.FILL_AND_STROKE);
886                         canvas.drawCircle((float) p.getX(),(float)
    p.getY(),DRAW_POINTSIZE/2,mPaintNo);
887                     }
888                     canvas.drawCircle((float) p.getX(),(float)
    p.getY(),DRAW_POINTSIZE/2,mPaintNo);
889
890                 //Desenho dos apoios
891                     apoioPath(E.mNos[i].getApNumero(), path, (float) mEscala/50);
892                     path.offset((float) p.getX(),(float) p.getY());
893                     canvas.drawPath(path, mPaintApoio);
894
895                 //Desenho das cargas pontuais
896                     if(E.mNos[i].getCp() != 0) {
897                         cPon = E.mCPon[E.mNos[i].getCp()];
898                         cargaPPath(E.mCPon[E.mNos[i].getCp()], path, (float) mEscala/70);
899                         path.offset((float) p.getX(),(float) p.getY());
900                         canvas.drawPath(path, mPaintPontual);
901
902                             if(cPon.getP(0) != 0) {
903                                 canvas.drawText(cPon.getP(0)+"
    "+getResources().getString(R.string.KN),
904                                         (float) (p.getX()-50*mEscala/70),(float)
    (p.getY()-8*mEscala/70), mPaintTexto);}
905                             if(cPon.getP(1) != 0) {
906                                 canvas.drawText(cPon.getP(1)+"
    "+getResources().getString(R.string.KN),
```

```
907                                         (float) (p.getX()+8*mEscala/70), (float)
    (p.getY()-30*mEscala/70), mPaintTexto);}
908                                 if(cPon.getP(2) != 0) {
909                                     canvas.drawText(cPon.getP(2)+"
    "+getResources().getString(R.string.KNm),
910                                         (float) (p.getX()+20*mEscala/70) ,(float)
    (p.getY()-10*mEscala/70), mPaintTexto);
911                                 }
912                             }
913                         }
914                     } else {
915                         for(int i = 0; i < E.mBarras.length; i++) {
916                             p = (E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)]).getC();
917                             q = (E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)]).getC();
918                             p = gridToTela(p);
919                             q = gridToTela(q);
920                             mPaintNo.setStyle(Style.STROKE);
921
922                             canvas.drawLine((float) p.getX(),(float) p.getY(),(float)
    q.getX(),(float) q.getY(), mPaintBarra);
923                             if(E.mBarras[i].getRot(Barra.NO_ESQUERDO) &&
    !E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)].getRot()) {
924                                 if(E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)].getSelection()) {
925                                     mPaintNo.setColor(getResources().getColor(R.color.BarraSelecio
    nada));
926                                 } else {
927                                     mPaintNo.setColor(getResources().getColor(R.color.No));
928                                 }
929                                 p.setX(p.getX()+DRAW_POINTSIZE*Math.cos(Math.toRadians(E.mBarras[i]
    .getAngulo())));
930                                 p.setY(p.getY()-DRAW_POINTSIZE*Math.sin(Math.toRadians(E.mBarras[i]
    .getAngulo())));
931                                 canvas.drawCircle((float) p.getX(),(float) p.getY(),
    DRAW_POINTSIZE, mEraser);
932                                 canvas.drawCircle((float) p.getX(),(float) p.getY(),
    DRAW_POINTSIZE, mPaintNo);
933                             }
934
935                             if(E.mBarras[i].getRot(Barra.NO_DIREITO) &&
    !E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)].getRot()) {
936                                 if(E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)].getSelection()) {
937                                     mPaintNo.setColor(getResources().getColor(R.color.BarraSelecio
    nada));
938                                 } else {
939                                 mPaintNo.setColor(getResources().getColor(R.color.No));
940                                 }
941                                 q.setX(q.getX()-DRAW_POINTSIZE*Math.cos(Math.toRadians(E.mBarras[i]
    .getAngulo())));
942                                 q.setY(q.getY()+DRAW_POINTSIZE*Math.sin(Math.toRadians(E.mBarras[i]
    .getAngulo())));
943                                 canvas.drawCircle((float) q.getX(),(float) q.getY(),
    DRAW_POINTSIZE, mEraser);
944                                 canvas.drawCircle((float) q.getX(),(float) q.getY(),
    DRAW_POINTSIZE, mPaintNo);
945                             }
946
947                             cDist = E.mCDis[E.mBarras[i].getQ()];
948                             switch(mResultados) {
949                             case RES_NORMAL : {
950                                 float[] r = new
    float[]{-E.mBarras[i].getEsforcosInternos(0).floatValue(),
```

```
951
    -E.mBarras[i].getEsforcosInternos(3).floatValue()};
952                         float l = (float) E.mBarras[i].getL();
953
954                         path.reset();
955                         path.moveTo((float) p.getX(), (float) p.getY());
956                         float f = 0;
957                         for(int j = 0; j <= nP; j++) {
958                             f = j/nP*l;
959                             x = (float) (p.getX() + f*mEscala);
960                             y = (float) (p.getY() - (r[0]-cDist.getQ(0)*(f-f*f/
    (2*l))-cDist.getQ(2)*(f*f/(2*l)))
961                                                                     *mEscala*mSeekBar.g
    etProgress()/2500);
962                             path.lineTo(x, y);
963                         }
964                         path.lineTo((float) (p.getX()+l*mEscala), (float) p.getY());
965                         canvas.save();
966                         canvas.rotate((float) (-E.mBarras[i].getAngulo()),(float)
    p.getX(),(float) p.getY());
967                         if(Math.abs(r[0]*Math.pow(10, 5)) >= 1) {
968                             mPaintTexto.setTextAlign(Align.LEFT);
969                             x = (float)  p.getX();
970                             y = (float) (p.getY() - r[0]
    *mEscala*mSeekBar.getProgress()/2500);
971                             canvas.drawText(format.format(r[0])+"kN", x, y-10,
    mPaintTexto);
972                         }
973                         if(Math.abs(r[1]*Math.pow(10, 5)) >= 1) {
974                             mPaintTexto.setTextAlign(Align.RIGHT);
975                             x = (float) (p.getX()+l*mEscala);
976                             y = (float) (p.getY() + r[1]
    *mEscala*mSeekBar.getProgress()/2500);
977                             canvas.drawText(format.format(-r[1])+"kN", x, y-10,
    mPaintTexto);
978                         }
979                         canvas.drawPath(path, mPaintDistribuida);
980                         canvas.restore();
981                         break;
982                         }
983                     case RES_CORTANTE : {
984
985                         float[] r = new
    float[]{E.mBarras[i].getEsforcosInternos(1).floatValue(),
986                                                 E.mBarras[i].getEsforcosInternos(4).floatV
    alue()};
987                         float l = (float) E.mBarras[i].getL();
988
989                         path.reset();
990                         path.moveTo((float) p.getX(), (float) p.getY());
991                         float f = 0;
992                         for(int j = 0; j <= nP; j++) {
993                             f = j/nP*l;
994                             x = (float) (p.getX() + f*mEscala);
995                             y = (float) (p.getY() - (r[0]+cDist.getQ(1)*(f-f*f/
    (2*l))+cDist.getQ(3)*(f*f/(2*l)))
996                                                                     *mEscala*mSeekBar.g
    etProgress()/2500);
997                             path.lineTo(x, y);
998                         }
999                         path.lineTo((float) (p.getX()+l*mEscala), (float) p.getY());
```

```
1000
1001                        canvas.save();
1002                        canvas.rotate((float) (-E.mBarras[i].getAngulo()),(float)
    p.getX(),(float) p.getY());
1003                        if(Math.abs(r[0]*Math.pow(10, 5)) >= 1) {
1004                            mPaintTexto.setTextAlign(Align.LEFT);
1005                            x = (float)  p.getX();
1006                            y = (float) (p.getY() - r[0]
    *mEscala*mSeekBar.getProgress()/2500);
1007                            canvas.drawText(format.format(r[0])+"kN", x, y-10,
    mPaintTexto);
1008                        }
1009                        if(Math.abs(r[1]*Math.pow(10, 5)) >= 1) {
1010                            mPaintTexto.setTextAlign(Align.RIGHT);
1011                            x = (float) (p.getX()+l*mEscala);
1012                            y = (float) (p.getY() + r[1]
    *mEscala*mSeekBar.getProgress()/2500);
1013                            canvas.drawText(format.format(-r[1])+"kN", x, y-10,
    mPaintTexto);
1014                        }
1015                        canvas.drawPath(path, mPaintDistribuida);
1016                        canvas.restore();
1017                        break;
1018                    }
1019                    case RES_MOMENTO : {
1020                        float[] r = new
    float[]{E.mBarras[i].getEsforcosInternos(1).floatValue(),
1021                                                E.mBarras[i].getEsforcosInternos(2).floatV
    alue(),
1022                                                E.mBarras[i].getEsforcosInternos(4).floatV
    alue(),
1023                                                E.mBarras[i].getEsforcosInternos(5).floatV
    alue()};
1024                        float l = (float) E.mBarras[i].getL();
1025
1026                        path.reset();
1027                        path.moveTo((float) p.getX(), (float) p.getY());
1028                        float f = 0;
1029                        for(int j = 0; j <= nP; j++) {
1030                            f = j/nP*l;
1031                            x = (float) (p.getX() + f*mEscala);
1032                            y = (float) (p.getY() - (r[1]-r[0]*f-cDist.getQ(1)*
    (f*f/2-f*f*f/(6*l))-cDist.getQ(3)*(f*f*f/(6*l)))
1033
    *mEscala*mSeekBar.getProgress()/2500);
1034                            path.lineTo(x, y);
1035                        }
1036                        path.lineTo((float) (p.getX()+l*mEscala), (float) p.getY());
1037
1038                        canvas.save();
1039                        canvas.rotate((float) (-E.mBarras[i].getAngulo()),(float)
    p.getX(),(float) p.getY());
1040                        if(Math.abs(r[1]*Math.pow(10, 5)) >= 1) {
1041                            mPaintTexto.setTextAlign(Align.LEFT);
1042                            x = (float)  p.getX();
1043                            y = (float) (p.getY() - r[1]
    *mEscala*mSeekBar.getProgress()/2500);
1044                            canvas.drawText(format.format(r[1])+"kNm", x, y-10,
    mPaintTexto);
1045                        }
1046                        if(Math.abs(r[3]*Math.pow(10, 5)) >= 1) {
```

```
1047                            mPaintTexto.setTextAlign(Align.RIGHT);
1048                            x = (float) (p.getX()+l*mEscala);
1049                            y = (float) (p.getY() + r[3]
    *mEscala*mSeekBar.getProgress()/2500);
1050                            canvas.drawText(format.format(-r[3])+"kNm", x, y-10,
    mPaintTexto);
1051                        }
1052                        canvas.drawPath(path, mPaintDistribuida);
1053                        canvas.restore();
1054                        break;
1055
1056                    }
1057                    case RES_DESLOCAMENTOS : {
1058
1059                        float[] d = new float[]{E.mBarras[i].getDesLocal(0).floatValue(),
1060                                                E.mBarras[i].getDesLocal(1).floatValue(),
1061                                                E.mBarras[i].getDesLocal(2).floatValue(),
1062                                                E.mBarras[i].getDesLocal(3).floatValue(),
1063                                                E.mBarras[i].getDesLocal(4).floatValue(),
1064                                                E.mBarras[i].getDesLocal(5).floatValue()};
1065                        float l = (float) E.mBarras[i].getL();
1066
1067                        float a = 0;
1068                        float b = 0;
1069                        float t = 0;
1070
1071                        path.reset();
1072                        path.moveTo((float) (p.getX() + d[0]
    *mEscala*mSeekBar.getProgress()*10), (float) (p.getY() - d[1]
    *mEscala*mSeekBar.getProgress()*10));
1073                        for(int j = 0; j <= nP; j++) {
1074                            t = j/nP;
1075                            a = d[2]*t*l+d[1]-d[4];
1076                            b = -(d[5]*t*l+d[1]-d[4]);
1077
1078                            x = (float) (p.getX() + t*l*mEscala + (d[0]+(d[3]-d[0])*t)
    *mEscala*mSeekBar.getProgress()*300000);
1079                            y = (float) (p.getY() - ((1-t)*d[1]+t*d[4]+t*(1-t)*(a*
    (1-t)+b*t))*mEscala*mSeekBar.getProgress()*300000);
1080
1081                            path.lineTo(x, y);
1082                            }
1083
1084                        canvas.save();
1085                        canvas.rotate((float) (-E.mBarras[i].getAngulo()),(float)
    p.getX(),(float) p.getY());
1086                        canvas.drawPath(path, mPaintDistribuida);
1087                        canvas.restore();
1088                        }
1089                    }
1090                }
1091
1092                for(int i = 0; i < E.mNos.length; i++) {
1093                    p = E.mNos[i].getC();
1094                    p = gridToTela(p);
1095                    canvas.drawCircle((float) p.getX(),(float)
    p.getY(),DRAW_POINTSIZE/2,mEraser);
1096                    if(E.mNos[i].getSelection()) {
1097                        mPaintNo.setColor(getResources().getColor(R.color.BarraSelecionada)
    );
1098                    } else {
```

```
1099                        mPaintNo.setColor(getResources().getColor(R.color.No));
1100                    }
1101                    if(E.mNos[i].getRot()) {
1102                        mPaintNo.setStyle(Style.STROKE);
1103                    } else {
1104                        mPaintNo.setStyle(Style.FILL_AND_STROKE);
1105                        canvas.drawCircle((float) p.getX(),(float)
    p.getY(),DRAW_POINTSIZE/2,mPaintNo);
1106                    }
1107                    canvas.drawCircle((float) p.getX(),(float)
    p.getY(),DRAW_POINTSIZE/2,mPaintNo);
1108
1109                //Desenho dos apoios
1110                    apoioPath(E.mNos[i].getApNumero(), path, (float) mEscala/50);
1111                    path.offset((float) p.getX(),(float) p.getY());
1112                    canvas.drawPath(path, mPaintApoio);
1113
1114                    if(mResultados == RES_REACOES) {
1115                        cPon = new CargaPontual(0, "", E.mNos[i].getRestricao(0)?
    E.mReacoesNodais.get(3*i, 0).doubleValue() : 0,
1116                                                E.mNos[i].getRestricao(1)?
    E.mReacoesNodais.get(3*i+1, 0).doubleValue() : 0,
1117                                                E.mNos[i].getRestricao(2)?
    E.mReacoesNodais.get(3*i+2, 0).doubleValue() : 0);
1118                        path.reset();
1119                        cargaPPath(cPon, path, (float) mEscala/70);
1120                        path.offset((float) p.getX(),(float) p.getY());
1121                        canvas.drawPath(path, mPaintPontual);
1122                        if(Math.abs(cPon.getP(0)*Math.pow(10, 10)) >= 1) {
1123                            canvas.drawText(format.format(cPon.getP(0))+"
    "+getResources().getString(R.string.KN),
1124                                    (float) (p.getX()-50*mEscala/70),(float)
    (p.getY()-8*mEscala/70), mPaintTexto);}
1125                        if(Math.abs(cPon.getP(1)*Math.pow(10, 10)) >= 1) {
1126                            canvas.drawText(format.format(cPon.getP(1))+"
    "+getResources().getString(R.string.KN),
1127                                    (float) (p.getX()+8*mEscala/70), (float)
    (p.getY()-30*mEscala/70), mPaintTexto);}
1128                        if(Math.abs(cPon.getP(2)*Math.pow(10, 10)) >= 1) {
1129                            canvas.drawText(format.format(cPon.getP(2))+"
    "+getResources().getString(R.string.KNm),
1130                                    (float) (p.getX()+20*mEscala/70) ,(float)
    (p.getY()-10*mEscala/70), mPaintTexto);
1131
1132                        }
1133                    }
1134                }
1135            }
1136        }
1137
1138        @Override
1139        public void invalidate() {
1140            super.invalidate();
1141        }
1142
1143        private class ScaleListener extends
    ScaleGestureDetector.SimpleOnScaleGestureListener {
1144
1145            @Override
1146            public boolean onScale(ScaleGestureDetector detector) {
1147                mEscala *= detector.getScaleFactor();
```

```
1148                invalidate();
1149                return true;
1150            }
1151        }
1152
1153        private class GestureListener extends GestureDetector.SimpleOnGestureListener {
1154
1155            Ponto orgInicial;
1156            Ponto orgFinal;
1157            Ponto escala;
1158
1159            @Override
1160            public boolean onDoubleTap(MotionEvent e) {
1161                if(E.mNos.length > 1) {
1162                Log.d("Ecalc","DoubleTap");
1163                RectF F = getRectEstrutura();
1164                escala = new Ponto(mEscala, Math.min(Math.abs(mSize.getX()/F.width()),
1165                                                     Math.abs(mSize.getY()/F.height())));
1166                orgInicial = new Ponto(mOrg);
1167                orgFinal = new Ponto(F.centerX(),F.centerY());
1168
1169                //mEscala = escala.getY();
1170                //mOrg = new Ponto(orgFinal);
1171
1172
1173                ValueAnimator vAnimator =
    ValueAnimator.ofFloat(0,1f).setDuration(ANIMATION_DURATION);
1174                vAnimator.setInterpolator(new OvershootInterpolator());
1175                vAnimator.addUpdateListener(new AnimatorUpdateListener() {
1176
1177                    @Override
1178                    public void onAnimationUpdate(ValueAnimator animation) {
1179                        mEscala = escala.getX() + (escala.getY()-escala.getX())
    *animation.getAnimatedFraction();
1180                        mOrg =
    orgInicial.soma((orgFinal.subtrai(orgInicial)).mult(animation.getAnimatedFraction()));
1181                        invalidate();
1182                    }
1183                });
1184                vAnimator.start();
1185                }
1186                return true;
1187            }
1188
1189            @Override
1190            public void onLongPress(MotionEvent e) {
1191                if(mState.getValue() == STATE_NORMAL) {
1192                    Ponto Ev = new Ponto(e.getX(),e.getY());
1193                    if(getNoSelecionado(Ev) == true) {
1194                        mState.setValue(STATE_SELECIONANO);
1195                    } else {
1196                        if(getBarraSelecionada(Ev) == true) {
1197                            mState.setValue(STATE_SELECIONABARRA);
1198                        }
1199                    }
1200                    invalidate();
1201                }
1202            }
1203        }
1204    }
1205
```

```
1206    //Configura o detector de clique para o menu
1207        private class SlideMenuClickListener implements
1208            ListView.OnItemClickListener {
1209            @Override //Quando um item é clicado
1210                public void onItemClick(AdapterView<?> parent, View view, int position,
1211                                        long id) {
1212                    //Chama a função
1213                    displayView(position);
1214            }
1215        }
1216
1217        //Quando cria o menu, cria a view
1218        public boolean onCreateOptionsMenu(Menu menu) {
1219            getMenuInflater().inflate(R.menu.main, menu);
1220            return true;
1221        }
1222
1223        @Override
1224        public boolean onOptionsItemSelected(MenuItem item) {
1225            // toggle nav drawer on selecting action bar app icon/title
1226            if (mDrawerToggle.onOptionsItemSelected(item)) {
1227                return true;
1228            }
1229            // Handle action bar actions click
1230            switch (item.getItemId()) {
1231            case R.id.action_settings:
1232                return true;
1233            default:
1234                return super.onOptionsItemSelected(item);
1235            }
1236        }
1237
1238        @Override
1239        public boolean onPrepareOptionsMenu(Menu menu) {
1240            // if nav drawer is opened, hide the action items
1241            boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerList);
1242            menu.findItem(R.id.action_settings).setVisible(!drawerOpen);
1243            return super.onPrepareOptionsMenu(menu);
1244        }
1245
1246        //Esta função é chamada toda vez que um item do menu é clicado. Serve para criar
1247        //os novos diálogos (Perfis, Cargas Distribuídas, etc.)
1248        private void displayView(int position) {
1249            Log.d("DisplayView", "Fragment");
1250            // update the main content by replacing fragments
1251            Fragment fragment = null;
1252            switch (position) {
1253            case 0:
1254                fragment = new Activity_DistribuidaTab();
1255                fragContainer.getLayoutParams().width = (int) (275*densidade);
1256                mState.setValue(STATE_FRAGMENT);
1257                break;
1258            case 1:
1259                fragment = new Activity_PontualTab();
1260                fragContainer.getLayoutParams().width = (int) (250*densidade);
1261                mState.setValue(STATE_FRAGMENT);
1262                break;
1263            case 2:
1264                fragment = new Activity_ApoioTab();
1265                fragContainer.getLayoutParams().width = (int) (200*densidade);
1266                mState.setValue(STATE_FRAGMENT);
```

```
1267                break;
1268            case 3:
1269                fragment = new Activity_PerfilTab();
1270                fragContainer.getLayoutParams().width = (int) (250*densidade);
1271                mState.setValue(STATE_FRAGMENT);
1272                break;
1273            case 4:
1274                fragment = new Activity_MaterialTab();
1275                fragContainer.getLayoutParams().width = (int) (250*densidade);
1276                mState.setValue(STATE_FRAGMENT);
1277                break;
1278            case 5:
1279                try {
1280                    E.calcular();
1281                    mState.setValue(STATE_RESULTADOS);
1282                    mCanvasView.postInvalidate();
1283                } catch(Exception e) {
1284                    Toast.makeText(getBaseContext(), e.getLocalizedMessage(),
    Toast.LENGTH_LONG).show();
1285                    e.printStackTrace();
1286                }
1287                break;
1288            case 6 :
1289                //TODO Implement exportation
1290                break;
1291            default:
1292                break;
1293            }
1294            if(fragment != null) {
1295                FragmentManager fragmentManager = getFragmentManager();
1296                fragmentManager.beginTransaction()
1297                        .add(R.id.frame_container,
    fragment).addToBackStack(null).commit();
1298            }
1299                //Atualiza o item e fecha o menu
1300                mDrawerList.setItemChecked(position, true);
1301                mDrawerList.setSelection(position);
1302                setTitle(navMenuTitles[position]);
1303                mDrawerLayout.closeDrawer(mDrawerList);
1304        }
1305
1306        @Override
1307        protected void onPostCreate(Bundle savedInstanceState) {
1308            super.onPostCreate(savedInstanceState);
1309            // Sync the toggle state after onRestoreInstanceState has occurred.
1310            mDrawerToggle.syncState();
1311        }
1312
1313        @Override
1314        public void onConfigurationChanged(Configuration newConfig) {
1315            super.onConfigurationChanged(newConfig);
1316            // Pass any configuration change to the drawer toggls
1317            mDrawerToggle.onConfigurationChanged(newConfig);
1318        }
1319
1320    //Muda o sistema da coordenadas da tela para o da E
1321    private Ponto telaToGrid(Ponto Pt) {
1322        Ponto Pg = new Ponto(Pt);
1323        Pg.setX(Pg.getX()-mSize.getX()/2);
1324        Pg.setY(-Pg.getY()+mSize.getY()/2);
1325        Pg = Pg.div(mEscala);
```

```
1326        Pg = Pg.soma(mOrg);
1327        return Pg;
1328    }
1329
1330    //Muda o sistema de coordenadas da E para o da tela
1331    private Ponto gridToTela(Ponto Pg) {
1332        Ponto Pt = new Ponto(Pg);
1333        Pt = Pt.subtrai(mOrg);
1334        Pt = Pt.mult(mEscala);
1335        Pt.setX(Pt.getX()+mSize.getX()/2);
1336        Pt.setY(-Pt.getY()+mSize.getY()/2);
1337        return Pt;
1338    }
1339
1340    private RectF getRectEstrutura() {
1341        Ponto Pmin, Pmax;
1342        try {
1343            Pmin = new Ponto(E.mNos[0].getC().getX(),E.mNos[0].getC().getY());
1344            Pmax = new Ponto(Pmin);}
1345        catch(Exception e) {
1346                return new RectF(0f,0f,0f,0f);
1347            }
1348        for(int i = 0; i < E.mNos.length; i++) {
1349            if(E.mNos[i].getC().getX() < Pmin.getX())
    {Pmin.setX(E.mNos[i].getC().getX());} else {
1350                if(E.mNos[i].getC().getX() > Pmax.getX())
    {Pmax.setX(E.mNos[i].getC().getX());}
1351            }
1352            if(E.mNos[i].getC().getY() < Pmin.getY())
    {Pmin.setY(E.mNos[i].getC().getY());} else {
1353                if(E.mNos[i].getC().getY() > Pmax.getY())
    {Pmax.setY(E.mNos[i].getC().getY());}
1354            }
1355        }
1356        Log.d("Ecalc",Pmin+" "+Pmax);
1357        return new RectF((float) (Pmin.getX()-1),(float) (Pmin.getY()-1),
1358                         (float) (Pmax.getX()+1),(float) (Pmax.getY()+1));
1359    }
1360
1361    private Path apoioPath(int n, Path p, float e) {
1362        p.reset();
1363        switch(n) {
1364        case 0 : break;
1365        case 1 : {
1366            p.lineTo(-e*20, e*10);
1367            p.lineTo(-e*20,-e*10);
1368            p.lineTo(    0,    0);
1369            p.moveTo(-e*25,-e*10);
1370            p.lineTo(-e*25, e*10);
1371        } break;
1372        case 2 : {
1373            p.lineTo( e*10, e*20);
1374            p.lineTo(-e*10, e*20);
1375            p.lineTo(    0,    0);
1376            p.moveTo(-e*10, e*25);
1377            p.lineTo( e*10, e*25);
1378        } break;
1379        case 3 : {
1380            p.lineTo( e*10, e*20);
1381            p.lineTo(-e*10, e*20);
1382            p.lineTo(    0,    0);
```

```
1383            p.moveTo(-e*10, e*25);
1384            p.lineTo( -e*5, e*20);
1385            p.lineTo(    0, e*25);
1386            p.lineTo(  e*5, e*20);
1387            p.lineTo( e*10, e*25);
1388        } break;
1389        case 4 : {
1390            p.addRect(-10*e, -10*e, 10*e, 10*e, Direction.CW);
1391            p.moveTo(0, 0);
1392            p.addRect(-5*e, -5*e, 5*e, 5*e, Direction.CW);
1393        } break;
1394        case 5 : {
1395            p.lineTo(    0,-10*e);
1396            p.lineTo( -5*e,-10*e);
1397            p.lineTo(    0, -5*e);
1398            p.lineTo( -5*e,    0);
1399            p.lineTo(    0,  5*e);
1400            p.lineTo( -5*e, 10*e);
1401            p.lineTo(    0, 10*e);
1402            p.lineTo(    0,    0);
1403            p.moveTo(-10*e,-10*e);
1404            p.lineTo(-10*e, 10*e);
1405        } break;
1406        case 6 : {
1407            p.lineTo(-e*10,    0);
1408            p.lineTo(-e*10,  e*5);
1409            p.lineTo( -e*5,    0);
1410            p.lineTo(    0,  e*5);
1411            p.lineTo(  e*5,    0);
1412            p.lineTo( e*10,  e*5);
1413            p.lineTo( e*10,    0);
1414            p.lineTo(    0,    0);
1415            p.moveTo(-e*10, 10*e);
1416            p.lineTo( e*10, 10*e);
1417        } break;
1418        case 7 : {
1419            p.lineTo(-e*10,   0);
1420            p.lineTo(-e*10, e*5);
1421            p.lineTo( -e*5,   0);
1422            p.lineTo(    0, e*5);
1423            p.lineTo(  e*5,   0);
1424            p.lineTo( e*10, e*5);
1425            p.lineTo( e*10,   0);
1426            p.lineTo(    0 ,  0);
1427        } break;
1428        }
1429        return p;
1430    }
1431
1432    private Path cargaPPath(CargaPontual cP, Path p, float e) {
1433        p.reset();
1434        //Desenho da carga em x
1435            if(Math.abs(cP.getP(0)*Math.pow(10,10)) >= 1) {//Existe carga nessa direção
1436                p.moveTo(-e*5, 0);
1437                p.lineTo(-e*40, 0);
1438                //Desenho da seta
1439                if(cP.getP(0) > 0) {//Seta está apontando para a direita
1440                    p.moveTo(-e*5,     0);
1441                    p.lineTo(-e*15, -e*3);
1442                    p.lineTo(-e*15,  e*3);
1443                    p.lineTo(-e*5,     0);
```

```
1444                } else {//Seta está apontando para a esquerda
1445                    p.lineTo(-e*30, -e*3);
1446                    p.lineTo(-e*30,  e*3);
1447                    p.lineTo(-e*40,    0);
1448                }
1449            }
1450        //Desenho da carga em y
1451            if(Math.abs(cP.getP(1)*Math.pow(10,10)) >= 1) {//Existe carga nessa direção
1452                p.moveTo(0, -e*5);
1453                p.lineTo(0, -e*40);
1454                //Desenho da seta
1455                if(cP.getP(1) > 0) {//Seta está apontando para cima
1456                    p.lineTo( -e*3, -e*30);
1457                    p.lineTo(  e*3, -e*30);
1458                    p.lineTo(    0, -e*40);
1459                } else {//Seta está apontando para baixo
1460                    p.moveTo(    0, -e*5);
1461                    p.lineTo( -e*3,-e*15);
1462                    p.lineTo( e*3, -e*15);
1463                    p.lineTo(   0,  -e*5);
1464                }
1465            }
1466        //Desenho momento em  z
1467            if(Math.abs(cP.getP(2)*Math.pow(10,10)) >= 1) {//Existe carga nessa direção
1468                p.moveTo(0,0);
1469                RectF oval = new RectF(-20*e,-20*e,20*e,20*e);
1470                p.addArc(oval, 180, 180);
1471                if(cP.getP(2) > 0) {//Seta está girando antihorário
1472                    p.moveTo(-20*e, 0);
1473                    p.lineTo(-23*e, 0);
1474                    p.lineTo(-20*e, 10*e);
1475                    p.lineTo(-17*e, 0);
1476                    p.lineTo(-20*e, 0);
1477                } else {//Seta está girando horário
1478                    p.lineTo( 23*e, 0);
1479                    p.lineTo( 20*e, 10*e);
1480                    p.lineTo( 17*e, 0);
1481                    p.lineTo( 20*e, 0);
1482                }
1483            }
1484        return p;
1485    }
1486
1487    public void changeState(int state) {
1488        mState.setValue(state);
1489    }
1490
1491    public void setApoio(int apoioId) {
1492        mSelec = SELEC_SETAPOIO;
1493        mState.setValue(STATE_SELECIONANO);
1494        fragInformation = apoioId;
1495    }
1496
1497    public void setPontual(int cPon) {
1498        mSelec = SELEC_SETPONTUAL;
1499        mState.setValue(STATE_SELECIONANO);
1500        fragInformation = cPon;
1501    }
1502
1503    public void setDistribuida(int cDis) {
1504        mSelec = SELEC_SETDISTRIBUIDA;
```

```
1505        mState.setValue(STATE_SELECIONABARRA);
1506        fragInformation = cDis;
1507    }
1508
1509    public void setPerfil(int perfil) {
1510        mSelec = SELEC_SETPERFIL;
1511        mState.setValue(STATE_SELECIONABARRA);
1512        fragInformation = perfil;
1513    }
1514
1515    public void setMaterial(int material) {
1516        mSelec = SELEC_SETMATERIAL;
1517        mState.setValue(STATE_SELECIONABARRA);
1518        fragInformation = material;
1519    }
1520
1521    public boolean getBarraSelecionada(Ponto Ev) {
1522        Log.d("Ecalc","Barra Selecionada");
1523        Ponto nEsquerdo;
1524        Ponto nDireito;
1525        Ponto maisProxima = new Ponto(-1,10000);
1526        for(int i = 0; i < E.mBarras.length; i++) {
1527            Log.d("Ecalc","Calculo da Barra "+i);
1528            nEsquerdo = E.mNos[E.mBarras[i].getN(Barra.NO_ESQUERDO)].getC();
1529            nDireito = E.mNos[E.mBarras[i].getN(Barra.NO_DIREITO)].getC();
1530            nEsquerdo = gridToTela(nEsquerdo);
1531            nDireito = gridToTela(nDireito);
1532            if(Ev.getDistancia(nDireito.soma(nEsquerdo).div(2)) < E.mBarras[i].getL()
    *mEscala/2) {
1533                double d = Ponto.distReta(nEsquerdo, nDireito, Ev);
1534                Log.d("Ecalc","Id = "+i+" , d="+d);
1535                if(d < maisProxima.getY()) {
1536                    maisProxima.setX(i);
1537                    maisProxima.setY(d);
1538                }
1539            }
1540        }
1541        if(maisProxima.getY() <= DISTANCIAMAXIMA) {
1542            E.mBarras[(int) maisProxima.getX()].setSelection(!E.mBarras[(int)
    maisProxima.getX()].getSelection());
1543            return true;
1544        }
1545        return false;
1546    }
1547
1548    public boolean getNoSelecionado(Ponto Ev) {
1549        Ponto maisProximo = new Ponto(-1,10000);
1550        for(int i = 0; i < E.mNos.length; i++) {
1551            double d = Ev.getDistancia(gridToTela(E.mNos[i].getC()));
1552            if(d < maisProximo.getY()) {
1553                maisProximo.setX(i);
1554                maisProximo.setY(d);
1555            }
1556        }
1557        if(maisProximo.getY() <= DISTANCIAMAXIMA) {
1558            E.mNos[(int) maisProximo.getX()].setSelection(!E.mNos[(int)
    maisProximo.getX()].getSelection());
1559            return true;
1560        }
1561        return false;
1562    }
```

```
1563
1564    public ArrayList<Integer> getNSelec() {
1565        ArrayList<Integer> nSelec = new ArrayList<Integer>();
1566        for(int i = 0; i < E.mNos.length; i++) {
1567            if(E.mNos[i].getSelection()) {
1568                nSelec.add(i);
1569            }
1570        }
1571        return nSelec;
1572    }
1573
1574    public ArrayList<Integer> getBSelec() {
1575        ArrayList<Integer> bSelec = new ArrayList<Integer>();
1576        for(int i = 0; i < E.mBarras.length; i++) {
1577            if(E.mBarras[i].getSelection()) {
1578                bSelec.add(i);
1579            }
1580        }
1581        return bSelec;
1582    }
1583
1584    public Ponto getGrid() {
1585        return mGrid;
1586    }
1587
1588    public void setGrid(Ponto grid) {
1589        mGrid = new Ponto(grid);
1590    }
1591
1592    public void updateMenuIndexes() {
1593        adapter.setCount(0, E.mCDis.length-1, true);
1594        adapter.setCount(1, E.mCPon.length-1, true);
1595        adapter.setCount(3, E.mPerfis.length-1, true);
1596        adapter.setCount(4, E.mMateriais.length-1, true);
1597        mDrawerList.setAdapter(adapter);
1598    }
1599 }
```

APÊNDICE F – Código dos Itens XML

F.1 Leiaute Carga Pontual

activity_pontualtab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity="bottom"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+id/ptl_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        >

      <TextView
        android:id="@+id/ptl_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pontual"
        android:textSize="20dp"
        android:layout_centerVertical="true" />

      <ImageButton
         android:id="@+id/ptl_btn_ret"
         android:layout_height="32dp"
         android:layout_width="32dp"
         android:layout_alignParentRight="true"
         android:layout_alignParentTop="true"
         android:background="@drawable/btn_return"
         />

     <ImageButton
        android:id="@+id/ptl_btn_add"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_marginRight="10dp"
        android:layout_toLeftOf="@+id/ptl_btn_ret"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_addelemento" />

     <ImageButton
        android:id="@+id/ptl_btn_exc"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_toLeftOf="@+id/ptl_btn_add"
        android:layout_marginRight="10dp"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_excluielemento" />


    </RelativeLayout>

     <TextView
         android:id="@+id/ptl_lbl_pontual"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentLeft="true"
```

```
62          android:layout_below="@id/ptl_view"
63          android:layout_marginTop="5dp"
64          android:text="@string/pontual_label" />
65
66      <TextView
67          android:id="@+id/ptl_lbl_nome"
68          android:layout_width="wrap_content"
69          android:layout_height="wrap_content"
70          android:layout_alignParentLeft="true"
71          android:layout_below="@+id/ptl_lbl_pontual"
72          android:layout_marginTop="5dp"
73          android:text="@string/nome_label" />
74
75      <TextView
76          android:id="@+id/ptl_lbl_fx"
77          android:layout_width="wrap_content"
78          android:layout_height="wrap_content"
79          android:layout_alignParentLeft="true"
80          android:layout_below="@+id/ptl_lbl_nome"
81          android:layout_marginTop="15dp"
82          android:text="@string/fx_label" />
83
84       <TextView
85          android:id="@+id/ptl_lbl_fy"
86          android:layout_width="wrap_content"
87          android:layout_height="wrap_content"
88          android:layout_alignParentLeft="true"
89          android:layout_below="@+id/ptl_lbl_fx"
90          android:layout_marginTop="5dp"
91          android:text="@string/fy_label" />
92
93      <TextView
94          android:id="@+id/ptl_lbl_mz"
95          android:layout_width="wrap_content"
96          android:layout_height="wrap_content"
97          android:layout_alignParentLeft="true"
98          android:layout_below="@+id/ptl_lbl_fy"
99          android:layout_marginTop="5dp"
100         android:text="@string/mz_label" />
101
102     <TextView
103         android:id="@+id/ptl_uni_fx"
104         android:layout_width="wrap_content"
105         android:layout_height="wrap_content"
106         android:layout_alignParentRight="true"
107         android:layout_alignBaseline="@id/ptl_lbl_fx"
108         android:layout_marginTop="5dp"
109         android:text="@string/KN" />
110
111     <TextView
112         android:id="@+id/ptl_uni_fy"
113         android:layout_width="wrap_content"
114         android:layout_height="wrap_content"
115         android:layout_alignParentRight="true"
116         android:layout_alignBaseline="@id/ptl_lbl_fy"
117         android:layout_marginTop="5dp"
118         android:text="@string/KN" />
119
120     <TextView
121         android:id="@+id/ptl_uni_mz"
122         android:layout_width="wrap_content"
```

```
123          android:layout_height="wrap_content"
124          android:layout_alignParentRight="true"
125          android:layout_alignBaseline="@id/ptl_lbl_mz"
126          android:layout_marginTop="5dp"
127          android:text="@string/KNm" />
128
129      <Spinner
130          android:id="@+id/ptl_spn_pontual"
131          android:layout_width="wrap_content"
132          android:layout_height="wrap_content"
133          android:layout_alignBottom="@+id/ptl_lbl_pontual"
134          android:layout_alignParentRight="true"
135          android:layout_alignTop="@+id/ptl_lbl_pontual"
136          android:layout_toRightOf="@+id/ptl_lbl_pontual"
137          android:layout_marginLeft="5dp" />
138
139      <ImageButton
140          android:id="@+id/ptl_btn_atualizar"
141          android:layout_width="32dp"
142          android:layout_height="32dp"
143          android:layout_alignParentRight="true"
144          android:background="@drawable/btn_atelemento"
145          android:layout_below="@id/ptl_spn_pontual"/>
146
147      <EditText
148          android:id="@+id/ptl_frm_nome"
149          android:layout_width="wrap_content"
150          android:layout_height="wrap_content"
151          android:layout_alignBaseline="@id/ptl_lbl_nome"
152          android:layout_marginLeft="5dp"
153          android:layout_toLeftOf="@+id/ptl_btn_atualizar"
154          android:layout_toRightOf="@id/ptl_lbl_pontual"
155          android:gravity="right"
156          android:maxLength="20"
157          android:maxLines="1"
158          android:textSize="12dp" />
159
160      <EditText
161         android:id="@+id/ptl_frm_fx"
162         android:layout_width="wrap_content"
163         android:layout_height="wrap_content"
164         android:layout_alignBaseline="@id/ptl_lbl_fx"
165         android:layout_toLeftOf="@id/ptl_uni_fx"
166         android:layout_toRightOf="@id/ptl_lbl_pontual"
167         android:layout_marginLeft="5dp"
168         android:textSize="12dp"
169         android:inputType="numberSigned|numberDecimal"
170         android:gravity="right" />
171
172      <EditText
173         android:id="@+id/ptl_frm_fy"
174         android:layout_width="wrap_content"
175         android:layout_height="wrap_content"
176         android:layout_alignBaseline="@id/ptl_lbl_fy"
177         android:layout_toLeftOf="@id/ptl_uni_fy"
178         android:layout_toRightOf="@id/ptl_lbl_pontual"
179         android:layout_marginLeft="5dp"
180         android:textSize="12dp"
181         android:inputType="numberSigned|numberDecimal"
182         android:gravity="right" />
183
```

```
184     <EditText
185        android:id="@+id/ptl_frm_mz"
186        android:layout_width="wrap_content"
187        android:layout_height="wrap_content"
188        android:layout_alignBaseline="@id/ptl_lbl_mz"
189        android:layout_toLeftOf="@id/ptl_uni_mz"
190        android:layout_toRightOf="@id/ptl_lbl_pontual"
191        android:layout_marginLeft="5dp"
192        android:textSize="12dp"
193        android:inputType="numberSigned|numberDecimal"
194        android:gravity="right" />
195
196     <ImageButton
197         android:id="@+id/ptl_btn_set"
198         android:layout_width="32dp"
199         android:layout_height="32dp"
200         android:layout_alignParentRight="true"
201         android:layout_below="@id/ptl_lbl_mz"
202         android:layout_marginTop="10dp"
203         android:background="@drawable/btn_set" />
204
205     <ImageButton
206         android:id="@+id/ptl_btn_setall"
207         android:layout_width="32dp"
208         android:layout_height="32dp"
209         android:layout_toLeftOf="@id/ptl_btn_set"
210         android:layout_below="@id/ptl_lbl_mz"
211         android:layout_marginTop="10dp"
212         android:layout_marginRight="10dp"
213         android:background="@drawable/btn_setall" />
214
215 </RelativeLayout>
216
217
218
```

F.2 Leiaute Carga Distribuída

activity_distribuidatab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:orientation="vertical"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity = "bottom" >

    <RelativeLayout
        android:id="@+id/dis_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        >
    <TextView
        android:id="@+id/dis_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/distribuida"
        android:textSize="20dp"
        android:layout_centerVertical="true"/>

    <ImageButton
        android:id="@+id/dis_btn_ret"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_return"/>

    <ImageButton
        android:id="@+id/dis_btn_add"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_marginRight="10dp"
        android:layout_toLeftOf="@id/dis_btn_ret"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_addelemento" />

    <ImageButton
        android:id="@+id/dis_btn_exc"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_alignParentTop="true"
        android:layout_marginRight="10dp"
        android:layout_toLeftOf="@+id/dis_btn_add"
        android:background="@drawable/btn_excluielemento" />

    </RelativeLayout>

    <TextView
        android:id="@+id/dis_lbl_distribuida"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/dis_view"
        android:layout_marginTop="5dp"
        android:text="@string/distribuida_label" />
```

```
62
63      <TextView
64          android:id="@+id/dis_lbl_nome"
65          android:layout_width="wrap_content"
66          android:layout_height="wrap_content"
67          android:layout_alignParentLeft="true"
68          android:layout_below="@+id/dis_lbl_distribuida"
69          android:layout_marginTop="17dp"
70          android:text="@string/nome_label" />
71
72      <TextView
73          android:id="@+id/dis_lbl_qx1"
74          android:layout_width="wrap_content"
75          android:layout_height="wrap_content"
76          android:layout_below="@id/dis_lbl_nome"
77          android:layout_marginLeft="5dp"
78          android:layout_marginTop="5dp"
79          android:text="@string/qx1_label" />
80
81      <TextView
82          android:id="@+id/dis_lbl_qy1"
83          android:layout_width="wrap_content"
84          android:layout_height="wrap_content"
85          android:layout_below="@id/dis_lbl_qx1"
86          android:layout_marginLeft="5dp"
87          android:layout_marginTop="5dp"
88          android:text="@string/qy1_label" />
89
90      <TextView
91          android:id="@+id/dis_lbl_qx2"
92          android:layout_width="wrap_content"
93          android:layout_height="wrap_content"
94          android:layout_below="@id/dis_lbl_qy1"
95          android:layout_marginLeft="5dp"
96          android:layout_marginTop="5dp"
97          android:text="@string/qx2_label" />
98
99      <TextView
100         android:id="@+id/dis_lbl_qy2"
101         android:layout_width="wrap_content"
102         android:layout_height="wrap_content"
103         android:layout_below="@id/dis_lbl_qx2"
104         android:layout_marginLeft="5dp"
105         android:layout_marginTop="5dp"
106         android:text="@string/qy2_label" />
107
108     <TextView
109         android:id="@+id/dis_uni_qx1"
110         android:layout_width="wrap_content"
111         android:layout_height="wrap_content"
112         android:layout_alignBaseline="@id/dis_lbl_qx1"
113         android:layout_alignParentRight="true"
114         android:layout_marginLeft="5dp"
115         android:layout_marginTop="5dp"
116         android:text="@string/KNDm" />
117
118     <TextView
119         android:id="@+id/dis_uni_qy1"
120         android:layout_width="wrap_content"
121         android:layout_height="wrap_content"
122         android:layout_alignBaseline="@id/dis_lbl_qy1"
```

```
123         android:layout_alignParentRight="true"
124         android:layout_marginLeft="5dp"
125         android:layout_marginTop="5dp"
126         android:text="@string/KNDm" />
127
128     <TextView
129         android:id="@+id/dis_uni_qx2"
130         android:layout_width="wrap_content"
131         android:layout_height="wrap_content"
132         android:layout_alignBaseline="@id/dis_lbl_qx2"
133         android:layout_alignParentRight="true"
134         android:layout_marginLeft="5dp"
135         android:layout_marginTop="5dp"
136         android:text="@string/KNDm" />
137
138     <TextView
139         android:id="@+id/dis_uni_qy2"
140         android:layout_width="wrap_content"
141         android:layout_height="wrap_content"
142         android:layout_alignBaseline="@id/dis_lbl_qy2"
143         android:layout_alignParentRight="true"
144         android:layout_marginLeft="5dp"
145         android:layout_marginTop="5dp"
146         android:text="@string/KNDm" />
147
148     <ImageButton
149         android:id="@+id/dis_btn_atualizar"
150         android:layout_width="32dp"
151         android:layout_height="32dp"
152         android:layout_alignBottom="@id/dis_lbl_nome"
153         android:layout_alignParentRight="true"
154         android:background="@drawable/btn_atelemento" />
155
156     <EditText
157         android:id="@+id/dis_frm_nome"
158         android:layout_width="wrap_content"
159         android:layout_height="wrap_content"
160         android:layout_alignBaseline="@id/dis_lbl_nome"
161         android:layout_marginLeft="5dp"
162         android:layout_toLeftOf="@id/dis_btn_atualizar"
163         android:layout_toRightOf="@id/dis_lbl_nome"
164         android:gravity="right"
165         android:maxLines="1"
166         android:textSize="12dp" />
167
168     <EditText
169         android:id="@+id/dis_frm_qx1"
170         android:layout_width="wrap_content"
171         android:layout_height="wrap_content"
172         android:layout_alignBaseline="@id/dis_lbl_qx1"
173         android:layout_marginLeft="5dp"
174         android:layout_toLeftOf="@id/dis_uni_qx1"
175         android:layout_toRightOf="@id/dis_lbl_qx1"
176         android:gravity="right"
177         android:inputType="numberSigned|numberDecimal"
178         android:maxLines="1"
179         android:textSize="12dp" />
180
181     <EditText
182         android:id="@+id/dis_frm_qy1"
183         android:layout_width="wrap_content"
```

```
184         android:layout_height="wrap_content"
185         android:layout_alignBaseline="@id/dis_lbl_qy1"
186         android:layout_marginLeft="5dp"
187         android:layout_toLeftOf="@id/dis_uni_qy1"
188         android:layout_toRightOf="@id/dis_lbl_qy1"
189         android:gravity="right"
190         android:inputType="numberSigned|numberDecimal"
191         android:maxLines="1"
192         android:textSize="12dp" />
193
194     <EditText
195         android:id="@+id/dis_frm_qx2"
196         android:layout_width="wrap_content"
197         android:layout_height="wrap_content"
198         android:layout_alignBaseline="@id/dis_lbl_qx2"
199         android:layout_marginLeft="5dp"
200         android:layout_toLeftOf="@id/dis_uni_qx2"
201         android:layout_toRightOf="@id/dis_lbl_qx2"
202         android:gravity="right"
203         android:inputType="numberSigned|numberDecimal"
204         android:maxLines="1"
205         android:textSize="12dp" />
206
207     <EditText
208         android:id="@+id/dis_frm_qy2"
209         android:layout_width="wrap_content"
210         android:layout_height="wrap_content"
211         android:layout_alignBaseline="@id/dis_lbl_qy2"
212         android:layout_marginLeft="5dp"
213         android:layout_toLeftOf="@id/dis_uni_qy2"
214         android:layout_toRightOf="@id/dis_lbl_qy2"
215         android:gravity="right"
216         android:inputType="numberSigned|numberDecimal"
217         android:textSize="12dp" />
218
219     <Spinner
220         android:id="@+id/dis_spn_distribuida"
221         android:layout_width="wrap_content"
222         android:layout_height="wrap_content"
223         android:layout_alignBottom="@+id/dis_lbl_distribuida"
224         android:layout_alignParentRight="true"
225         android:layout_alignTop="@+id/dis_lbl_distribuida"
226         android:layout_marginLeft="5dp"
227         android:layout_toRightOf="@+id/dis_lbl_distribuida" />
228
229     <RadioGroup
230         android:id="@+id/dis_rdg"
231         android:layout_width="fill_parent"
232         android:layout_height="wrap_content"
233         android:layout_alignParentLeft="true"
234         android:layout_below="@+id/dis_lbl_qy2"
235         android:layout_marginTop="10dp"
236         android:gravity="center_horizontal"
237         android:orientation="horizontal" >
238
239         <RadioButton
240             android:id="@+id/dis_btn_local"
241             android:text="@string/local_label" />
242
243         <RadioButton
244             android:id="@+id/dis_btn_global"
```

```
245             android:layout_marginLeft="10dp"
246             android:text="@string/global_label" />
247     </RadioGroup>
248 
249     <ImageButton
250         android:id="@+id/dis_btn_set"
251         android:layout_width="32dp"
252         android:layout_height="32dp"
253         android:layout_below="@id/dis_rdg"
254         android:layout_marginTop="10dp"
255         android:layout_alignParentRight="true"
256         android:background="@drawable/btn_set" />
257 
258     <ImageButton
259         android:id="@+id/dis_btn_setall"
260         android:layout_width="32dp"
261         android:layout_height="32dp"
262         android:layout_below="@id/dis_rdg"
263         android:layout_marginTop="10dp"
264         android:layout_marginRight="10dp"
265         android:layout_toLeftOf="@id/dis_btn_set"
266         android:background="@drawable/btn_setall" />
267 
268 </RelativeLayout>
```

F.3 Leiaute Material

activity_materialtab.xml

```
<?xml version="1.0" encoding="utf-8"?><RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:layout_gravity="bottom" >

    <!-- Título do Layout -->
    <RelativeLayout
        android:id="@+id/mat_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true">

    <TextView
        android:id="@+id/mat_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/material"
        android:textSize="20dp"
        android:layout_centerVertical="true"/>

    <ImageButton
        android:id="@+id/mat_btn_ret"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_return" />

     <ImageButton
        android:id="@+id/mat_btn_add"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_toLeftOf="@+id/mat_btn_ret"
        android:layout_alignParentTop="true"
        android:layout_marginRight="10dp"
        android:background="@drawable/btn_addelemento" />

     <ImageButton
        android:id="@+id/mat_btn_exc"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_toLeftOf="@+id/mat_btn_add"
        android:layout_marginRight="10dp"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_excluielemento" />

     </RelativeLayout>

    <!-- Rótulo Material -->
     <TextView
        android:id="@+id/mat_lbl_material"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/mat_view"
        android:text="@string/material_label"
```

```
61          android:layout_marginTop="5dp"/>
62
63      <TextView
64          android:id="@+id/mat_lbl_nome"
65          android:layout_width="wrap_content"
66          android:layout_height="wrap_content"
67          android:layout_alignParentLeft="true"
68          android:layout_below="@id/mat_lbl_material"
69          android:text="@string/nome_label"
70          android:layout_marginTop="15dp"/>
71
72       <ImageButton
73          android:id="@+id/mat_btm_atualizar"
74          android:layout_height="32dp"
75          android:layout_width="32dp"
76          android:layout_alignBottom="@+id/mat_lbl_nome"
77          android:layout_marginLeft="10dp"
78          android:layout_alignParentRight="true"
79          android:background="@drawable/btn_atelemento" />
80
81      <TextView
82          android:id="@+id/mat_lbl_elasticidade"
83          android:layout_width="wrap_content"
84          android:layout_height="wrap_content"
85          android:layout_alignParentLeft="true"
86          android:layout_below="@id/mat_lbl_nome"
87          android:text="@string/modE_label"
88          android:layout_marginTop="5dp"/>
89
90       <TextView
91          android:id="@+id/mat_uni_elasticidade"
92          android:layout_width="wrap_content"
93          android:layout_height="wrap_content"
94          android:layout_alignParentRight="true"
95          android:layout_alignBaseline="@id/mat_lbl_elasticidade"
96          android:text="@string/MPa"
97          android:layout_marginTop="5dp"/>
98
99       <TextView
100          android:id="@+id/mat_lbl_densidade"
101          android:layout_width="wrap_content"
102          android:layout_height="wrap_content"
103          android:layout_below="@+id/mat_lbl_elasticidade"
104          android:text="@string/dVol_label"
105          android:layout_marginTop="5dp"/>
106
107       <TextView
108          android:id="@+id/mat_uni_densidade"
109          android:layout_width="wrap_content"
110          android:layout_height="wrap_content"
111          android:layout_alignParentRight="true"
112          android:layout_alignBaseline="@id/mat_lbl_densidade"
113          android:text="@string/kgm3"
114          android:layout_marginTop="5dp"/>
115
116      <Spinner
117           android:id="@+id/mat_spn_material"
118           android:layout_width="0px"
119           android:layout_height="wrap_content"
120           android:layout_above="@+id/mat_lbl_nome"
121           android:layout_marginLeft="5dp"
```

```
122           android:layout_alignParentRight="true"
123           android:layout_alignTop="@+id/mat_lbl_material"
124           android:layout_toRightOf="@id/mat_lbl_material" />
125
126       <EditText
127         android:id="@+id/mat_frm_nome"
128         android:layout_height = "wrap_content"
129         android:layout_width = "0px"
130         android:layout_alignBaseline = "@id/mat_lbl_nome"
131         android:layout_toRightOf="@id/mat_lbl_nome"
132         android:layout_toLeftOf="@id/mat_btm_atualizar"
133         android:gravity="right"
134         android:maxLength="20"
135         android:textSize="12dp"
136         android:maxLines="1"  />
137
138        <EditText
139         android:id="@+id/mat_frm_elasticidade"
140         android:layout_height = "wrap_content"
141         android:layout_width = "0px"
142         android:layout_alignBaseline="@id/mat_lbl_elasticidade"
143         android:layout_toRightOf="@id/mat_lbl_elasticidade"
144         android:layout_toLeftOf="@id/mat_uni_elasticidade"
145         android:textSize="12dp"
146         android:gravity="right"
147         android:inputType="numberDecimal" />
148
149       <EditText
150         android:id="@+id/mat_frm_densidade"
151         android:layout_height = "wrap_content"
152         android:layout_width = "0px"
153         android:layout_toRightOf="@id/mat_lbl_densidade"
154         android:layout_toLeftOf="@id/mat_uni_densidade"
155         android:layout_alignBaseline="@id/mat_lbl_densidade"
156         android:textSize="12dp"
157         android:gravity="right"
158         android:inputType="numberDecimal"/>
159
160      <ImageButton
161         android:id="@+id/mat_btn_set"
162         android:layout_height="32dp"
163         android:layout_width="32dp"
164         android:layout_alignParentRight="true"
165         android:layout_below="@id/mat_lbl_densidade"
166         android:layout_marginTop="10dp"
167         android:background="@drawable/btn_set" />
168
169       <ImageButton
170           android:id="@+id/mat_btn_setall"
171           android:layout_width="32dp"
172           android:layout_height="32dp"
173           android:layout_toLeftOf="@+id/mat_btn_set"
174           android:layout_alignTop="@+id/mat_btn_set"
175           android:layout_marginRight="10dp"
176           android:background="@drawable/btn_setall" />
177
178 </RelativeLayout>
```

F.4 Leiaute Perfil

activity_perfiltab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity="bottom"
     >

    <RelativeLayout
        android:id="@+id/per_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true">

    <TextView
        android:id="@+id/per_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/perfil"
        android:textSize="20dp"
        android:layout_centerVertical="true" />

    <ImageButton
        android:id="@+id/per_btn_ret"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_return" />

    <ImageButton
        android:id="@+id/per_btn_add"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_marginRight="10dp"
        android:layout_toLeftOf="@+id/per_btn_ret"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_addelemento" />

    <ImageButton
        android:id="@+id/per_btn_exc"
        android:layout_height="32dp"
        android:layout_width="32dp"
        android:layout_toLeftOf="@+id/per_btn_add"
        android:layout_marginRight="10dp"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_excluielemento" />

    </RelativeLayout>

       <TextView
        android:id="@+id/per_lbl_perfil"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/per_view"
        android:text="@string/perfil_label"
        android:layout_marginTop="5dp"/>
```

```
62
63        <TextView
64         android:id="@+id/per_lbl_nome"
65         android:layout_width="wrap_content"
66         android:layout_height="wrap_content"
67         android:layout_alignParentLeft="true"
68         android:layout_below="@id/per_lbl_perfil"
69         android:text="@string/nome_label"
70         android:layout_marginTop="15dp"/>
71
72        <TextView
73         android:id="@+id/per_lbl_tipo"
74         android:layout_width="wrap_content"
75         android:layout_height="wrap_content"
76         android:layout_alignParentLeft="true"
77         android:layout_below="@id/per_lbl_nome"
78         android:text="@string/tipo_label"
79         android:layout_marginTop="5dp"/>
80
81      <LinearLayout
82          android:id="@+id/per_img_perfil"
83          android:layout_below="@+id/per_lbl_tipo"
84          android:layout_above="@+id/per_lbl_area"
85          android:layout_width="80dp"
86          android:layout_height="80dp"
87          android:layout_marginTop="10dp"
88          android:layout_alignParentLeft="true"
89          android:orientation="horizontal"/>
90
91        <TextView
92         android:id="@+id/per_lbl_d0"
93         android:layout_width="wrap_content"
94         android:layout_height="wrap_content"
95         android:layout_toRightOf="@id/per_img_perfil"
96         android:layout_below="@id/per_lbl_tipo"
97         android:text="d0:"
98         android:layout_marginLeft="5dp"
99         android:layout_marginTop="5dp"/>
100
101       <TextView
102        android:id="@+id/per_lbl_d1"
103        android:layout_width="wrap_content"
104        android:layout_height="wrap_content"
105        android:layout_toRightOf="@id/per_img_perfil"
106        android:layout_below="@id/per_lbl_d0"
107        android:text="d1:"
108        android:layout_marginLeft="5dp"
109        android:layout_marginTop="5dp"/>
110
111       <TextView
112        android:id="@+id/per_lbl_d2"
113        android:layout_width="wrap_content"
114        android:layout_height="wrap_content"
115        android:layout_toRightOf="@id/per_img_perfil"
116        android:layout_below="@id/per_lbl_d1"
117        android:text="d2:"
118        android:layout_marginLeft="5dp"
119        android:layout_marginTop="5dp"/>
120
121       <TextView
122        android:id="@+id/per_lbl_d3"
```

```
123         android:layout_width="wrap_content"
124         android:layout_height="wrap_content"
125         android:layout_toRightOf="@id/per_img_perfil"
126         android:layout_below="@id/per_lbl_d2"
127         android:text="d3:"
128         android:layout_marginLeft="5dp"
129         android:layout_marginTop="5dp"/>
130
131       <TextView
132         android:id="@+id/per_lbl_area"
133         android:layout_width="wrap_content"
134         android:layout_height="wrap_content"
135         android:layout_alignParentLeft="true"
136         android:layout_below="@id/per_lbl_d3"
137         android:text="@string/area_label"
138         android:layout_marginTop="5dp"/>
139
140       <TextView
141         android:id="@+id/per_lbl_inercia"
142         android:layout_width="wrap_content"
143         android:layout_height="wrap_content"
144         android:layout_alignParentLeft="true"
145         android:layout_below="@id/per_lbl_area"
146         android:text="@string/inercia_label"
147         android:layout_marginTop="5dp"/>
148
149       <TextView
150         android:id="@+id/per_uni_d0"
151         android:layout_width="wrap_content"
152         android:layout_height="wrap_content"
153         android:layout_alignParentRight="true"
154         android:layout_alignBaseline="@id/per_lbl_d0"
155         android:text="@string/mm"
156         android:layout_marginTop="5dp"/>
157
158       <TextView
159         android:id="@+id/per_uni_d1"
160         android:layout_width="wrap_content"
161         android:layout_height="wrap_content"
162         android:layout_alignParentRight="true"
163         android:layout_alignBaseline="@id/per_lbl_d1"
164         android:text="@string/mm"
165         android:layout_marginTop="5dp"/>
166
167       <TextView
168         android:id="@+id/per_uni_d2"
169         android:layout_width="wrap_content"
170         android:layout_height="wrap_content"
171         android:layout_alignParentRight="true"
172         android:layout_alignBaseline="@id/per_lbl_d2"
173         android:text="@string/mm"
174         android:layout_marginTop="5dp"/>
175
176       <TextView
177         android:id="@+id/per_uni_d3"
178         android:layout_width="wrap_content"
179         android:layout_height="wrap_content"
180         android:layout_alignParentRight="true"
181         android:layout_alignBaseline="@id/per_lbl_d3"
182         android:text="@string/mm"
183         android:layout_marginTop="5dp"/>
```

```
184
185         <TextView
186          android:id="@+id/per_uni_area"
187          android:layout_width="wrap_content"
188          android:layout_height="wrap_content"
189          android:layout_alignParentRight="true"
190          android:layout_alignBaseline="@id/per_lbl_area"
191          android:text="@string/mm2"
192          android:layout_marginTop="5dp"/>
193
194         <TextView
195          android:id="@+id/per_uni_inercia"
196          android:layout_width="wrap_content"
197          android:layout_height="wrap_content"
198          android:layout_alignParentRight="true"
199          android:layout_alignBaseline="@id/per_lbl_inercia"
200          android:text="@string/mm4"
201          android:layout_marginTop="5dp"/>
202
203         <Spinner
204             android:id="@+id/per_spn_perfil"
205             android:layout_width="wrap_content"
206             android:layout_height="wrap_content"
207             android:layout_alignBottom="@+id/per_lbl_perfil"
208             android:layout_alignParentRight="true"
209             android:layout_toRightOf="@id/per_lbl_tipo"
210             android:layout_alignTop="@+id/per_lbl_perfil"
211             android:layout_marginLeft="10dp"
212             />
213
214         <Spinner
215             android:id="@+id/per_spn_tipo"
216             android:layout_width="wrap_content"
217             android:layout_height="wrap_content"
218             android:layout_alignBottom="@+id/per_lbl_tipo"
219             android:layout_alignParentRight="true"
220             android:layout_toRightOf="@id/per_lbl_tipo"
221             android:layout_alignTop="@+id/per_lbl_tipo"
222             android:layout_marginLeft="10dp"
223             />
224
225         <ImageButton
226             android:id="@+id/per_btn_atualizar"
227             android:layout_width="32dp"
228             android:layout_height="32dp"
229             android:layout_alignParentRight="true"
230             android:background="@drawable/btn_atelemento"
231             android:layout_alignBottom="@id/per_lbl_nome"/>
232
233         <EditText
234             android:id="@+id/per_frm_nome"
235             android:layout_width="wrap_content"
236             android:layout_height="wrap_content"
237             android:layout_alignBaseline="@id/per_lbl_nome"
238             android:layout_toRightOf="@id/per_lbl_nome"
239             android:layout_toLeftOf="@id/per_btn_atualizar"
240             android:maxLength="20"
241             android:textSize="12dp"
242             android:gravity="right"
243             android:maxLines="1" />
244
```

```
245         <EditText
246             android:id="@+id/per_frm_d0"
247             android:layout_width="wrap_content"
248             android:layout_height="wrap_content"
249             android:layout_alignBaseline="@id/per_lbl_d0"
250             android:layout_toLeftOf="@id/per_uni_d0"
251             android:layout_toRightOf="@id/per_img_perfil"
252             android:layout_marginLeft="20dp"
253             android:textSize="12dp"
254             android:inputType="numberDecimal"
255             android:gravity="right"/>
256 
257         <EditText
258             android:id="@+id/per_frm_d1"
259             android:layout_width="wrap_content"
260             android:layout_height="wrap_content"
261             android:layout_alignBaseline="@id/per_lbl_d1"
262             android:layout_toRightOf="@id/per_img_perfil"
263             android:layout_marginLeft="20dp"
264             android:layout_toLeftOf="@id/per_uni_d1"
265             android:textSize="12dp"
266             android:inputType="numberDecimal"
267             android:gravity="right" />
268 
269         <EditText
270             android:id="@+id/per_frm_d2"
271             android:layout_width="wrap_content"
272             android:layout_height="wrap_content"
273             android:layout_alignBaseline="@id/per_lbl_d2"
274             android:layout_toRightOf="@id/per_img_perfil"
275             android:layout_marginLeft="20dp"
276             android:layout_toLeftOf="@id/per_uni_d2"
277             android:textSize="12dp"
278             android:inputType="numberDecimal"
279             android:gravity="right" />
280 
281         <EditText
282             android:id="@+id/per_frm_d3"
283             android:layout_width="wrap_content"
284             android:layout_height="wrap_content"
285             android:layout_alignBaseline="@id/per_lbl_d3"
286             android:layout_toRightOf="@id/per_img_perfil"
287             android:layout_marginLeft="20dp"
288             android:layout_toLeftOf="@id/per_uni_d3"
289             android:textSize="12dp"
290             android:inputType="numberDecimal"
291             android:gravity="right"/>
292 
293         <EditText
294             android:id="@+id/per_frm_area"
295             android:layout_width="wrap_content"
296             android:layout_height="wrap_content"
297             android:layout_alignBaseline="@id/per_lbl_area"
298             android:layout_toRightOf="@id/per_lbl_area"
299             android:layout_toLeftOf="@id/per_uni_area"
300             android:textSize="12dp"
301             android:inputType="numberDecimal"
302             android:gravity="right"
303             android:enabled="false" />
304 
305         <EditText
```

```
306             android:id="@+id/per_frm_inercia"
307             android:layout_width="wrap_content"
308             android:layout_height="wrap_content"
309             android:layout_alignBaseline="@id/per_lbl_inercia"
310             android:layout_toRightOf="@id/per_lbl_inercia"
311             android:layout_toLeftOf="@id/per_uni_inercia"
312             android:textSize="12dp"
313             android:inputType="numberDecimal"
314             android:gravity="right"
315             android:enabled="false" />
316
317     <ImageButton
318         android:id="@+id/per_btn_set"
319         android:layout_height="32dp"
320         android:layout_width="32dp"
321         android:layout_below="@id/per_lbl_inercia"
322         android:layout_alignParentRight="true"
323         android:layout_marginTop="10dp"
324         android:background="@drawable/btn_set" />
325
326     <ImageButton
327         android:id="@+id/per_btn_setall"
328         android:layout_height="32dp"
329         android:layout_width="32dp"
330         android:layout_below="@id/per_lbl_inercia"
331         android:layout_toLeftOf="@id/per_btn_set"
332         android:layout_marginRight="10dp"
333         android:layout_marginTop="10dp"
334         android:background="@drawable/btn_setall" />
335
336 </RelativeLayout>
337
```

F.5 Leiaute Apoio

activity_apoiotab.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:orientation="vertical"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity="bottom">

    <RelativeLayout
        android:id="@+id/apo_title_view"
        android:layout_width="match_parent"
        android:layout_height="32dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="7dp"
        >

        <TextView
            android:id="@+id/apo_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:text="@string/apoio"
            android:textSize="20dp"
            android:layout_centerVertical="true"/>

        <ImageButton
            android:id="@+id/apo_btn_ret"
            android:layout_width="32dp"
            android:layout_height="32dp"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:background="@drawable/btn_return" />

    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/apo_vie_apoio"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_alignLeft="@+id/apo_title_view"
        android:layout_below="@id/apo_title_view"
        android:layout_marginTop="10dp"
        >

        <ImageView
            android:id="@+id/apo_img_apoio"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/ap_001"
            android:layout_centerInParent="true"/>

    </RelativeLayout>

    <TextView
        android:id="@+id/apo_lbl_Rx"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below= "@id/apo_title_view"
```

```
62          android:layout_toRightOf="@id/apo_vie_apoio"
63          android:layout_marginLeft="10dp"
64          android:layout_marginTop="10dp"
65          android:text="@string/Rx_label"/>
66
67      <TextView
68          android:id="@+id/apo_lbl_Ry"
69          android:layout_width="wrap_content"
70          android:layout_height="wrap_content"
71          android:layout_below="@id/apo_lbl_Rx"
72          android:layout_toRightOf="@id/apo_vie_apoio"
73          android:layout_marginLeft="10dp"
74          android:layout_marginTop="5dp"
75          android:text="@string/Ry_label"/>
76
77      <TextView
78          android:id="@+id/apo_lbl_Rz"
79          android:layout_width="wrap_content"
80          android:layout_height="wrap_content"
81          android:layout_below="@id/apo_lbl_Ry"
82          android:layout_toRightOf="@id/apo_vie_apoio"
83          android:layout_marginLeft="10dp"
84          android:layout_marginTop="5dp"
85          android:text="@string/Rz_label"/>
86
87      <CheckBox
88          android:id="@+id/apo_btn_Rx"
89          android:layout_width="wrap_content"
90          android:layout_height="wrap_content"
91          android:layout_alignParentRight="true"
92          android:layout_alignBaseline="@id/apo_lbl_Rx" />
93
94      <CheckBox
95          android:id="@+id/apo_btn_Ry"
96          android:layout_width="wrap_content"
97          android:layout_height="wrap_content"
98          android:layout_alignParentRight="true"
99          android:layout_alignBaseline="@id/apo_lbl_Ry" />
100
101     <CheckBox
102         android:id="@+id/apo_btn_Rz"
103         android:layout_width="wrap_content"
104         android:layout_height="wrap_content"
105         android:layout_alignParentRight="true"
106         android:layout_alignBaseline="@id/apo_lbl_Rz" />
107
108     <ImageButton
109         android:id="@+id/apo_btn_set"
110         android:layout_width="32dp"
111         android:layout_height="32dp"
112         android:layout_marginTop="10dp"
113         android:layout_alignLeft="@+id/apo_btn_Rz"
114         android:layout_below="@+id/apo_lbl_Rz"
115         android:background="@drawable/btn_set" />
116
117     <ImageButton
118         android:id="@+id/apo_btn_setall"
119         android:layout_width="32dp"
120         android:layout_height="32dp"
121         android:layout_toLeftOf="@id/apo_btn_set"
122         android:layout_marginTop="10dp"
```

```
123         android:layout_marginRight="10dp"
124         android:layout_below="@id/apo_lbl_Rz"
125         android:background="@drawable/btn_setall"
126         />
127
128
129 </RelativeLayout>
```

F.6 Leiaute Nó

activity_notab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity="bottom"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+id/brs_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:gravity="center_vertical">

    <TextView
        android:id="@+id/brs_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="@string/nos"
        android:textSize="20dp"
        android:layout_centerVertical="true" />

    <ImageButton
        android:id="@+id/nos_btn_ret"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_return" />

    </RelativeLayout>

    <TextView
        android:id="@+id/nos_lbl_ptl"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/brs_view"
        android:layout_marginTop="5dp"
        android:text="@string/pontual_label" />

    <Spinner
        android:id="@+id/nos_spn_ptl"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/nos_lbl_ptl"
        android:layout_alignTop="@+id/nos_lbl_ptl"
        android:layout_marginLeft="10dp"
        android:layout_toRightOf="@+id/nos_lbl_ptl" />

    <TextView
        android:id="@+id/nos_lbl_rX"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
```

```
62          android:layout_below="@id/nos_spn_ptl"
63          android:layout_marginTop="5dp"
64          android:text="@string/Rx_label" />
65
66      <TextView
67          android:id="@+id/nos_lbl_rY"
68          android:layout_width="wrap_content"
69          android:layout_height="wrap_content"
70          android:layout_alignParentLeft="true"
71          android:layout_below="@+id/nos_lbl_rX"
72          android:layout_marginTop="5dp"
73          android:text="@string/Ry_label" />
74
75      <TextView
76          android:id="@+id/nos_lbl_rZ"
77          android:layout_width="wrap_content"
78          android:layout_height="wrap_content"
79          android:layout_alignParentLeft="true"
80          android:layout_below="@+id/nos_lbl_rY"
81          android:layout_marginTop="5dp"
82          android:text="@string/Rz_label" />
83
84      <CheckBox
85          android:id="@+id/nos_chb_rX"
86          android:layout_width="wrap_content"
87          android:layout_height="wrap_content"
88          android:layout_alignParentRight="true"
89          android:layout_alignBaseline="@id/nos_lbl_rX"
90          />
91
92      <CheckBox
93          android:id="@+id/nos_chb_rY"
94          android:layout_width="wrap_content"
95          android:layout_height="wrap_content"
96          android:layout_alignParentRight="true"
97          android:layout_alignBaseline="@id/nos_lbl_rY"
98           />
99
100     <CheckBox
101         android:id="@+id/nos_chb_rZ"
102         android:layout_width="wrap_content"
103         android:layout_height="wrap_content"
104         android:layout_alignParentRight="true"
105         android:layout_alignBaseline="@id/nos_lbl_rZ"
106         />
107
108     <CheckBox
109         android:id="@+id/nos_chb_rot"
110         android:layout_width="32dp"
111         android:layout_height="32dp"
112         android:layout_marginTop="10dp"
113         android:layout_alignParentRight="true"
114         android:layout_below="@id/nos_lbl_rZ"
115         android:button="@drawable/btn_rot"
116         />
117
118 </RelativeLayout>
```

F.7 Leiaute Barra

activity_barratab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity="bottom"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+id/brs_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:gravity="center_vertical">

    <TextView
        android:id="@+id/brs_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="@string/barras"
        android:textSize="20dp"
        android:layout_centerVertical="true"/>

     <ImageButton
        android:id="@+id/brs_btn_ret"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:background="@drawable/btn_return"/>

    </RelativeLayout>

    <TextView
        android:id="@+id/nos_lbl_ptl"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/brs_view"
        android:layout_marginTop="5dp"
        android:text="@string/perfil_label" />

    <TextView
        android:id="@+id/brs_lbl_mat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/nos_lbl_ptl"
        android:layout_marginTop="5dp"
        android:text="@string/material_label" />

    <TextView
        android:id="@+id/brs_lbl_dis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
```

```
62          android:layout_below="@id/brs_lbl_mat"
63          android:layout_marginTop="5dp"
64          android:text="@string/distribuida_label" />
65
66      <Spinner
67          android:id="@+id/brs_spn_per"
68          android:layout_width="wrap_content"
69          android:layout_height="wrap_content"
70          android:layout_marginLeft="10dp"
71          android:layout_alignBottom="@+id/nos_lbl_ptl"
72          android:layout_alignTop="@+id/nos_lbl_ptl"
73          android:layout_toRightOf="@+id/nos_lbl_ptl"
74          />
75
76      <Spinner
77          android:id="@+id/brs_spn_mat"
78          android:layout_width="wrap_content"
79          android:layout_height="wrap_content"
80          android:layout_marginLeft="10dp"
81          android:layout_alignBottom="@+id/brs_lbl_mat"
82          android:layout_alignTop="@+id/brs_lbl_mat"
83          android:layout_toRightOf="@+id/brs_lbl_mat"
84          />
85
86       <Spinner
87          android:id="@+id/brs_spn_dis"
88          android:layout_width="wrap_content"
89          android:layout_height="wrap_content"
90          android:layout_marginLeft="10dp"
91          android:layout_alignBottom="@+id/brs_lbl_dis"
92          android:layout_alignTop="@+id/brs_lbl_dis"
93          android:layout_toRightOf="@+id/brs_lbl_dis"
94          />
95
96      <CheckBox
97          android:id="@+id/brs_btn_rotD"
98          android:layout_width="32dp"
99          android:layout_height="32dp"
100         android:layout_marginTop="10dp"
101         android:layout_below="@id/brs_spn_dis"
102         android:layout_alignParentRight="true"
103         android:button="@drawable/btn_rotd" />
104
105     <CheckBox
106         android:id="@+id/brs_btn_rotE"
107         android:layout_width="32dp"
108         android:layout_height="32dp"
109         android:layout_marginTop="10dp"
110         android:layout_below="@id/brs_spn_dis"
111         android:layout_toLeftOf="@id/brs_btn_rotE"
112         android:button="@drawable/btn_rote"/>
113
114 </RelativeLayout>
115
116
117
```

F.8 Leiaute Grelha

activity_gridtab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:padding="5dp"
    android:background="@color/BackgroundFragment"
    android:gravity="top">

    <RelativeLayout
        android:id="@+id/grd_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        >

    <ImageButton
        android:id="@+id/grd_btn_ret"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:background="@drawable/btn_return" />

    <TextView
        android:id="@+id/grd_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="@string/grid"
        android:textSize="20dp"
        android:layout_centerVertical="true"/>

     </RelativeLayout>

    <TextView
        android:id="@+id/grd_lbl_x"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/grd_view"
        android:layout_marginTop="5dp"
        android:text="@string/x_label"/>

    <TextView
        android:id="@+id/grd_lbl_y"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="5dp"
        android:layout_below="@id/grd_lbl_x"
        android:text="@string/y_label"/>

    <TextView
        android:id="@+id/grd_uni_x"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_marginTop="5dp"
        android:layout_below="@id/grd_view"
```

```
62          android:text="@string/m"/>
63
64      <TextView
65          android:id="@+id/grd_uni_y"
66          android:layout_width="wrap_content"
67          android:layout_height="wrap_content"
68          android:layout_alignParentRight="true"
69          android:layout_marginTop="5dp"
70          android:layout_below="@id/grd_lbl_x"
71          android:text="@string/m"/>
72
73      <EditText
74          android:id="@+id/grd_edt_x"
75          android:layout_width="wrap_content"
76          android:layout_height="wrap_content"
77          android:layout_alignBaseline="@+id/grd_uni_x"
78          android:layout_toLeftOf="@id/grd_uni_x"
79          android:layout_toRightOf="@id/grd_lbl_x"
80          android:gravity="right"
81          android:inputType="numberDecimal"
82          android:digits="0123456789."
83          android:maxLength="20"
84          android:maxLines="1"
85          android:textSize="12dp" />
86
87      <EditText
88          android:id="@+id/grd_edt_y"
89          android:layout_width="wrap_content"
90          android:layout_height="wrap_content"
91          android:layout_alignBaseline="@+id/grd_uni_y"
92          android:layout_below="@+id/grd_lbl_x"
93          android:layout_toLeftOf="@id/grd_uni_y"
94          android:layout_toRightOf="@id/grd_lbl_y"
95          android:gravity="right"
96          android:inputType="numberDecimal"
97          android:digits="0123456789."
98          android:maxLength="20"
99          android:maxLines="1"
100         android:textSize="12dp" />
101
102 </RelativeLayout>
```

F.9 Leiaute Principal

activity_canvas.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true">


    <android.support.v4.widget.DrawerLayout
        android:id="@+id/drawer_layout"
        android:layout_width="200dp"
        android:layout_height="match_parent">

        <LinearLayout
            android:clickable="true"
            android:id="@+id/cnv_canvas"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"
            android:orientation="vertical"/>

        <ListView
            android:id="@+id/list_slidermenu"
            android:layout_width="10dp"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            android:choiceMode="singleChoice"
            android:divider="@color/list_divider"
            android:dividerHeight="0dp"
            android:listSelector="@drawable/list_selector"
            android:background="@color/list_background"
            android:gravity="center_vertical"/>

         <ImageView
            android:id="@+id/cnv_click"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true" />

    </android.support.v4.widget.DrawerLayout>

    <ImageButton
        android:id="@+id/btn_menu"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginRight="5dp"
        android:layout_marginBottom="5dp"
        android:background="@drawable/cog"
        />


     <ImageButton
        android:id="@+id/btn_exc"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_above="@id/btn_menu"
```

```
62          android:layout_alignParentRight="true"
63          android:layout_marginRight="5dp"
64          android:layout_marginBottom="5dp"
65          android:background="@drawable/exc"
66          />
67
68       <ImageButton
69          android:id="@+id/btn_addBarra"
70          android:layout_width="64dp"
71          android:layout_height="64dp"
72          android:layout_above="@id/btn_exc"
73          android:layout_alignParentRight="true"
74          android:layout_marginRight="5dp"
75          android:layout_marginBottom="5dp"
76          android:background="@drawable/barra_add"
77          />
78
79       <ImageButton
80          android:id="@+id/btn_apply"
81          android:layout_width="64dp"
82          android:layout_height="64dp"
83          android:layout_alignParentBottom="true"
84          android:layout_alignParentRight="true"
85          android:layout_marginRight="5dp"
86          android:layout_marginBottom="5dp"
87          android:background="@drawable/cog"
88          />
89
90        <ImageButton
91          android:id="@+id/btn_ret"
92          android:layout_width="64dp"
93          android:layout_height="64dp"
94          android:layout_above="@id/btn_apply"
95          android:layout_alignParentRight="true"
96          android:layout_marginRight="5dp"
97          android:layout_marginBottom="5dp"
98          android:background="@drawable/ret"
99          />
100
101       <ImageButton
102          android:id="@+id/btn_grid"
103          android:layout_width="64dp"
104          android:layout_height="64dp"
105          android:layout_alignParentTop="true"
106          android:layout_alignParentRight="true"
107          android:layout_marginRight="5dp"
108          android:layout_marginTop="5dp"
109          android:background="@drawable/ret"
110          />
111
112       <FrameLayout
113          android:id="@+id/frame_container"
114          android:layout_width="200dp"
115          android:layout_height="wrap_content"
116          android:layout_alignParentRight="true"
117          android:layout_alignParentBottom="true"
118          android:layout_marginRight="5dp"
119          android:layout_marginBottom="5dp"
120          android:gravity="bottom"
121       />
122
```

```
123 </RelativeLayout>
124
125
126
```

APÊNDICE G – Código XML complementares

G.1 Valores de texto

strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Ecalc</string>
    <string name="action_settings">Settings</string>

    <!-- Titles -->
        <string name="material">MATERIAIS</string>
        <string name="perfil">PERFIS</string>
        <string name="pontual">PONTUAIS</string>
        <string name="distribuida">DISTRIBUÍDAS</string>
        <string name="apoio">APOIOS</string>
        <string name="barras">BARRAS</string>
        <string name="nos">NÓS</string>
        <string name="grid">GRELHA</string>

    <!-- Labels -->
        <string name="altura_label">h</string>
        <string name="area_label">A</string>
        <string name="base_label">b</string>
        <!-- Diametros -->
            <string name="diametro_label">&#xD8;</string>
            <string
  name="diametro_externo_label">&#xD8;<inf><small>e</small></inf></string>
        <!-- Espessuras -->
            <string name="espessura_label">t</string>
            <string name="espessura_alma_label">t<inf><small>w</small></inf></string>
            <string name="espessura_mesa_label">t<inf><small>f</small></inf></string>
        <string name="distribuida_label">Distribuída</string>
        <string name="dVol_label">&#961;</string>
        <string name="fx_label">F<small><inf>x</inf></small></string>
        <string name="fy_label">F<small><inf>y</inf></small></string>
        <string name="global_label">Global</string>
        <string name="inercia_label">I</string>
        <string name="local_label">Local</string>
        <string name="material_label">Material</string>
        <string name="modE_label">E</string>
        <string name="mz_label">M<small><inf>z</inf></small></string>
        <string name="nome_label">Nome</string>
        <string name="perfil_label">Perfil</string>
        <string name="pontual_label">Pontual</string>
        <string name="qx1_label">q<small><inf>x1</inf></small></string>
        <string name="qx2_label">q<small><inf>x2</inf></small></string>
        <string name="qy1_label">q<small><inf>y1</inf></small></string>
        <string name="qy2_label">q<small><inf>y2</inf></small></string>
        <string name="Rx_label">R<inf><small>x</small></inf></string>
        <string name="Ry_label">R<inf><small>y</small></inf></string>
        <string name="Rz_label">R<inf><small>z</small></inf></string>
        <string name="tipo_label">Tipo</string>
        <string name="x_label">X</string>
        <string name="y_label">Y</string>

    <!-- Exceptions -->
        <string name="materialexception">Há elementos sem material definido</string>
        <string name="perfilexception">Há elementos sem perfil definido</string>
        <string name="hipostaticaexception">A estrutura está hipostática</string>
        <string name="unknownexception">Exceção desconhecida</string>

    <!-- Unidades -->
        <string name="MPa">MPa</string>
        <string name="mm">mm</string>
```

```
61          <string name="mm2">mm<sup><small>2</small></sup></string>
62          <string name="mm4">mm<sup><small>4</small></sup></string>
63          <string name="m">m</string>
64          <string name="kgm3">Kg/m&#179;</string>
65          <string name="KN">KN</string>
66          <string name="KNm">KNm</string>
67          <string name="KNDm">KN/m</string>
68
69      <!-- Menu -->
70          <string-array name="nav_drawer_items">
71              <item>Distribuídas</item>
72              <item>Pontuais</item>
73              <item>Apoios</item>
74              <item>Perfis</item>
75              <item>Materiais</item>
76              <item>Calcular</item>
77              <item>Exportar</item>
78         </string-array>
79
80         <array name="nav_drawer_icons">
81              <item>@drawable/per_icn_1</item>
82              <item>@drawable/per_icn_2</item>
83              <item>@drawable/per_icn_3</item>
84              <item>@drawable/per_icn_4</item>
85              <item>@drawable/per_icn_6</item>
86              <item>@drawable/per_icn_7</item>
87              <item>@drawable/per_icn_7</item>
88         </array>
89         <string name="desc_list_item_icon">Item Icon</string>
90
91      <!-- Perfis -->
92          <string name="perfil_generico">Genérico</string>
93          <string name="perfil_circular">Circular</string>
94          <string name="perfil_retangular">Retangular</string>
95          <string name="perfil_circularvazado">Circular Vazado</string>
96          <string name="perfil_retangularvazado">Retangular Vazado</string>
97          <string name="perfil_i">Perfil I</string>
98          <string name="perfil_t">Perfil T</string>
99
100     <!-- Toasts Mensagens -->
101         <string name="dimensaoinvalida_toast">Dimensões Inválidas. Considerar
    Revisão.</string>
102         <string name="nome_invalido">Nome Inválido</string>
103 </resources>
104
```

G.2 Valores de cor

color.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="Battleship_Grey">#848482</color>
    <color name="Chestnut_Red">#C34A2C</color>
    <color name="Denim_Blue">#79BAEC</color>
    <color name="Gray_Cloud">#B6B6B4</color>
    <color name="Mahogany">#C04000</color>
    <color name="Night">#0C090A</color>
    <color name="Slate_Blue">#737CA1</color>
    <color name="BackgroundFragment">#E5E4E2</color>

    <color name="BarraNormal">#347C2C</color>
    <color name="BarraSelecionada">#254117</color>
    <color name="No">#C34A2C</color>
    <color name="CargaPontual">#79BAEC</color>
    <color name="Grid">#B6B5B4</color>
    <color name="CargaDistribuida">#737CA1</color>
    <color name="Texto">#0C090A</color>
    <color name="Apoio">#848482</color>

    <color name="list_divider">#0C090A</color>
    <color name="list_background">#B6B6B4</color>
    <color name="list_item_title">#E5E4E2</color> <!-- Platinum -->
    <color name="counter_text_color">#728C00</color> <!-- Venom Green -->
    <color name="counter_text_bg">#E5E4E2</color> <!-- Platinum -->
    <color name="list_background_pressed">#728C00</color>
</resources>

```